Exhibit

Proceedings of the 11th Conference (1979 International) on Solid State Devices, Tokyo, 1979;
Japanese Journal of Applied Physics, Volume 19 (1980) Supplement 19-1, pp. 193-202

Adaptive Wafer Scale Integration

Yukun HSIA, Gareth C. C. CHANG and F. Dennis ERWIN

McDonnell Douglas Astronautics Company,
Huntington Beach, CA 92647, USA

Based on an in-situ electrical alterable nonvolatile semiconductor memory, the MNOS transistor, an adaptive interconnect is developed to implement wafer-scale integration. Experimental validation of the interconnect is reported. The interconnect concept is further extended to the design of semiconductor mass memory and the design of an adaptive voter to implement fault tolerant systems.

§1. Introduction

Adaptive Wafer Scale Integration (AWSI) is a concept which originated in late 1971 at Actron,* a division of McDonnell Douglas Corporation.[1] It employs electrically alterable, nonvolatile interconnect controller circuits processed into a semiconductor wafer to connect "arrays" of interconnected, operable circuits (also processed into the wafer) to a bus structure deposited on the wafer between the arrays. With this approach, AWSI can have important advantages relative to other high density electronic circuit approaches. Such advantages potentially include: repeated electronic reconfigurability of interconnected circuits, compatibility with a wide variety of semiconductor processes, ability to select for yield, improved reliability, self-healing and fault tolerance; all plus reduction in size, weight, and cost.

Figure 1 graphically illustrates the AWSI concept in contrast with conventional integrated circuit (IC) technology. In both technologies the "chips" or "dice" (called arrays in AWSI) are probed immediately after the completion of wafer processing to test for operability. In conventional IC technology, those dice which do not pass the probe test

*Now a part of the McDonnell Douglas Astronautics Company.

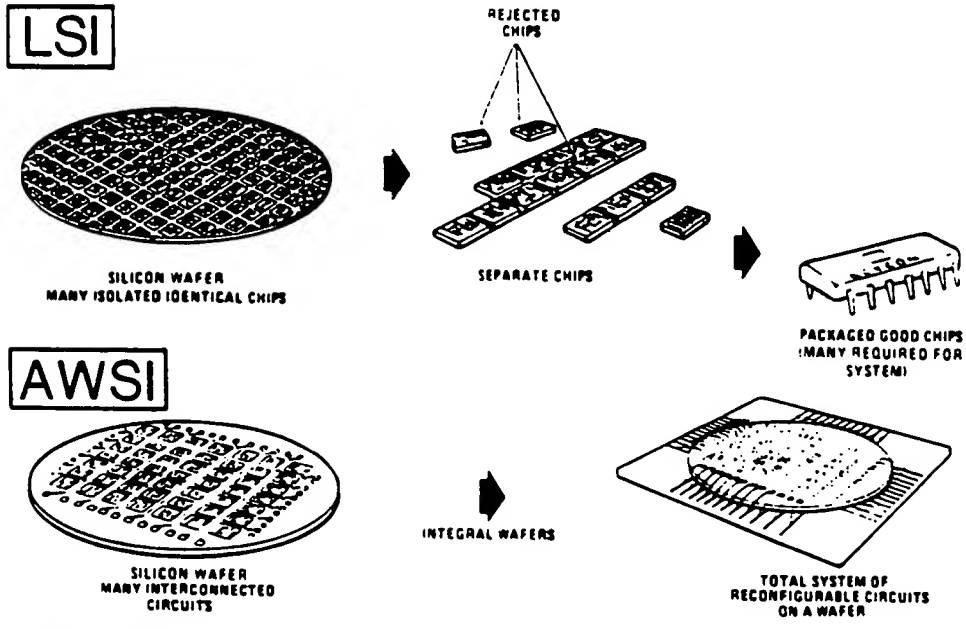


Fig. 1. Conventional integrated circuit packaging versus wafer-scale integration packaging.

are marked and discarded when the wafer is diced into individual chips.

In AWSI, the die array are all connected via system bus. The address of these arrays that pass the probe test are stored in nonvolatile memory. Operable arrays which are initially used in system mechanization are accessible either via indirect addressing or via associative decoder addressing. The remainder of the operable arrays are reserved as spares to be connected into the bus structure as replacements for operating arrays which subsequently may be shown by built-in tests to have become defective. Alternatively, the spares can be later used for system reconfiguration, dependent on application requirements.

The use of a repeatedly alterable, electrical interconnect, processed into the wafer provides the principal advantages of AWSI compared to other whole-wafer technologies.[2-6] With AWSI interconnect

a. Any array may be connected to, or disconnected from, the structure at any time; connections include both signals and power.
b. Neither nonoperable nor nonoperating arrays draw power.
c. Neither special contact masks, metalization masks nor fusible links are required to connect arrays to bus lines. Conditional interconnect is accomplished electronically instead of via nonalterable mechanical means.
d. Spare operable arrays are stored on the wafer and connected to the bus structure whenever built-in tests reveal that one of the active arrays has failed and must be replaced.
e. Reconfiguration may be achieved within limits established via the system architecture by reconnecting arrays to the bus.

§2. Basis for AWSI

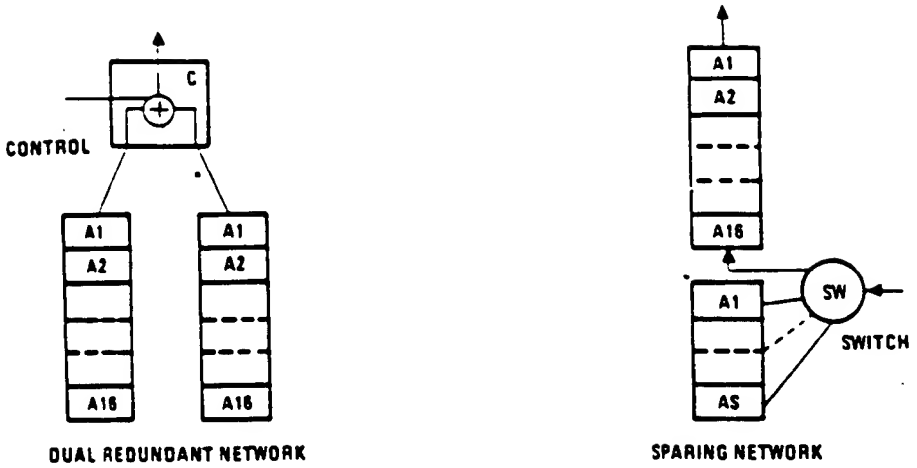
The technological basis for the AWSI interconnect is the nonvolatile semiconductor memory transistor, the MNOS (metal-nitride-oxide-semiconductor) transistor.[7-9]

The MNOS transistor is unique among solid-state electronic devices in that it permits stored data alterability simultaneously with stored data nonvolatility.[10-11] It also provides a small-area structure and fabrication compatibility with high density, integrated circuit technologies.

The MNOS transistor is a close relative of the conventional metal-oxide-semiconductor (MOS) transistor in that the usual layer of gate oxide is replaced by a 500 Å layer of silicon nitride over a less than 20 Å thick layer of silicon dioxide. The application of a moderately-high voltage (approximately 25 volts) to the gate electrode of this transistor causes the thin silicon dioxide layer to become conductive (or to permit charge carriers to tunnel through it). Charge carriers may then pass between the silicon and charge-carrier traps located near the silicon-nitride, silicon-dioxide interface. The presence of trapped charge carriers at this interface modifies the gate voltage which controls passage of charge carriers from source to drain in the conventional operation of the transistor. The MNOS transistor is then said to have an "off" and an "on" states which depend on the concentration and the polarity of the trapped charge carriers.

The MNOS transistors, utilized in a circuit, effectively became the means with which electrically alterable interconnection can be implemented in place of hardwired interconnections. The alterable interconnect can be so utilized because the interconnection has a nonvolatile memory for status control so that the state of interconnected machine is static as if hardwired. Yet it is alterable, (that is adaptive), via control signals by circuit means.

The adaptive feature of the interconnect makes it possible to implement sparing to produce high reliability memory systems. A minimum number of spare memory circuits is set aside on the wafer. Upon detection of failure, the failed component is disconnected and the spare is connected to the system bus. Utilizing sparing redundancy, high system reliability is accomplished with minimum hardware cost and little increase in parts count. An illustrative example is given in Fig. 2. Assume 16 memory circuits interconnected to form a high reliability memory subsystem, assigning same failure rates to equivalent-function units, it can be seen that using two spares for a 16 unit serial memory system, an order of magnitude improvement in calculated failure rate is achieved over that utilizing the dual-redundant approach. The improvement is impressive since



RELIABILITY	0.9144/5 YRS	RELIABILITY	0.98858/5 YRS FOR 2 SPARES
FAILURE RATE	0.2%/1000 HRS	FAILURE RATE	0.026%/1000 HRS
		RELIABILITY	0.98989/5 YRS FOR 4 SPARES
		FAILURE RATE	0.023%/1000 HRS

Fig. 2. Example of gain in reliability with the use of sparing redundancy.

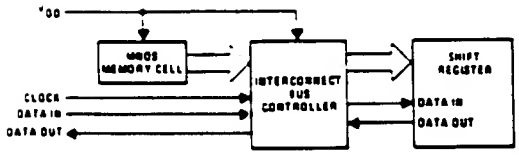


Fig. 3. Use of AWSI to interconnect a shift register string.

the sparing redundancy is realized with reduced hardware requirements and minimum operating power.

As an introduction to the AWSI concepts in concrete physical terms, we can use the example of an early feasibility demonstration vehicle, the reconfigurable memory wafer circuits, RMWC-1. The interconnect is composed of an electrically alterable, nonvolatile, MNOS memory cell and a standard logic or buffer gate NMOS transistor circuit or bus controller. The bus controller is enabled by the enabling signal which is conditional on the stored interconnect status of the memory cell. The bus controller connects or gates the power, clock and data signals from the interconnecting bus to the engaged circuit, in this case, a shift register. The relationship of the MNOS memory cell, the bus controller and the shift register in the AWSI implementation is shown in Fig. 3.

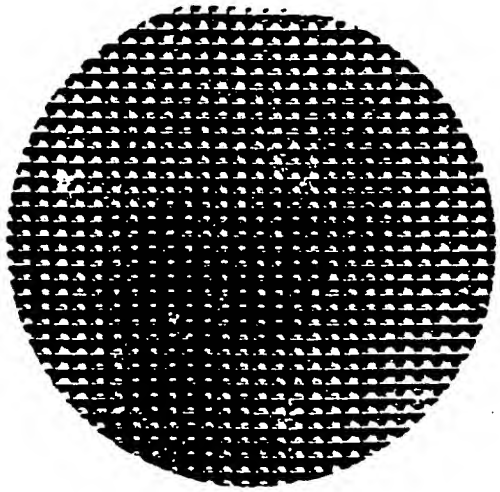


Fig. 4. Use of AWSI to interconnect shift register strings on a wafer, (the RMWC-1).

This particular circuit has been fabricated and integrated on a wafer, Fig. 4; the data path organization is illustrated in Fig. 5. It may be seen that a shift register in a given horizontal string may be either connected in series with the data line or bypassed. Each repeated combination is designated as an array. The metalization pattern for this wafer circuit is shown in Fig. 6.

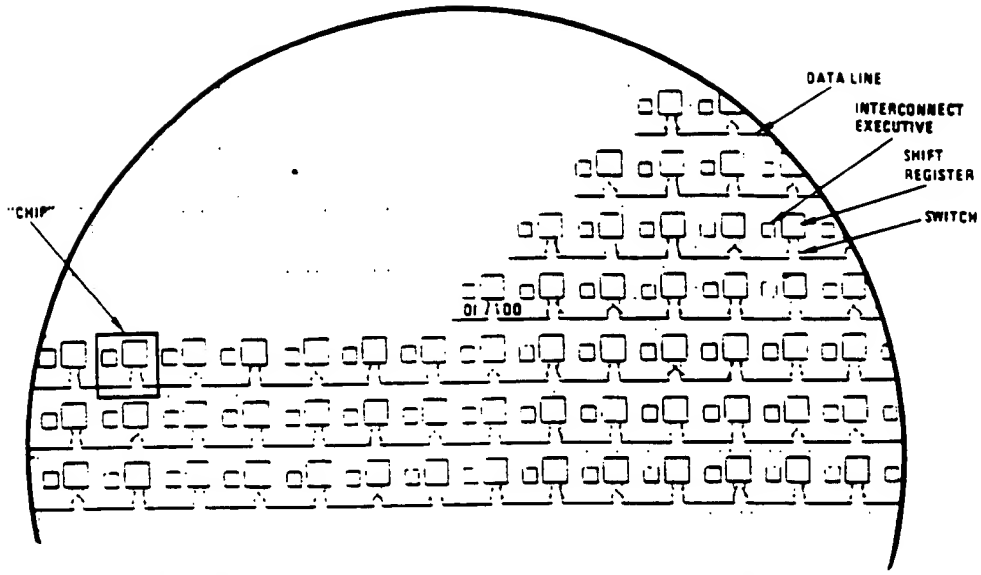


Fig. 5. Data path organization of RMWC-1.

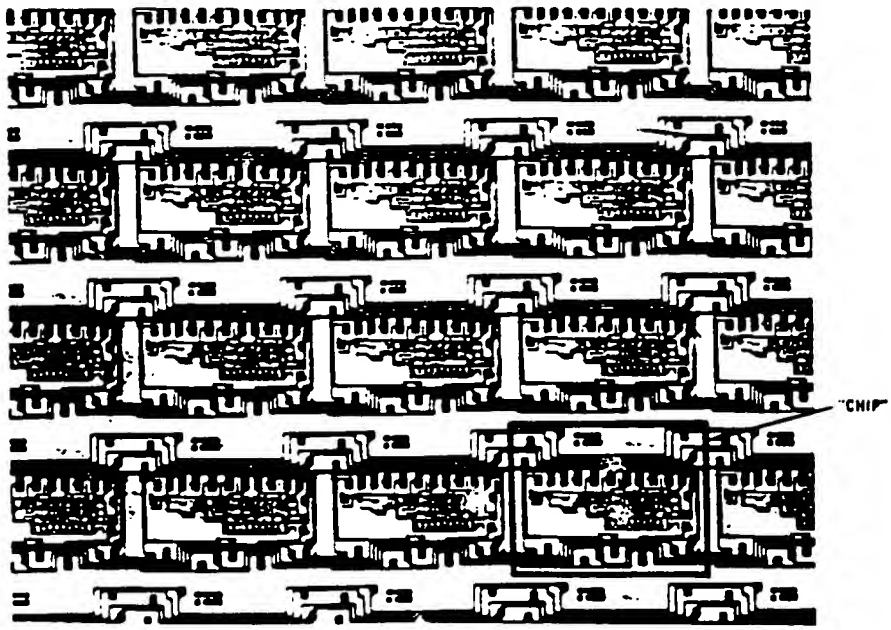


Fig. 6. The metalization pattern of RMWC-1.

In the nonvolatile mass memory system, the application of AWSI is further extended. On-the-wafer interconnect is mechanized such that small single-chip sequential access memory arrays (with memory storage capacity in the range of 4 K to 16 K bits per memory array) are interconnected to result in a large capacity store (on the order of 10^6 bits) in a single 4 inch wafer. The use of nonvolatile memories as the storage array in a mass memory system results in very low system powwer as compared with other memories implemented with RAM's or CCD's. Power need be applied only to those storage arrays accessed, with no standby power necessary to maintain data in unaccessed storage arrays. Additionally, with the use of AWSI sparing redundancy, the mass memory achieves system reliability at low cost.

§3. Memory System Hardware

The memory system can be divided into wafer memory modules and the off-wafer circuitry, Fig. 7. The main components of the off-wafer circuitry are the system controller and the error detection/correction circuitry. The wafer memory modules consist of memory wafers, which in turn consist of the channel controllers (CC), the array controllers (AC), the memory arrays (MA), and the buses which connect them, Fig. 8.

The off-wafer system controller supplies the intelligence of the system. Not only does the system controller interpret the system commands and supply detailed instruction sequencing to the wafers, it also monitors the operation of all the system components. Should a fault be detected, the system controller will execute a built-in-test to isolate the fault, and then reconfigure the system to remove the fault.

The off-wafer error detection/correction circuitry performs the encoding/deconding and error correction on the system data. In conjunction with the system buffer, the error detection/

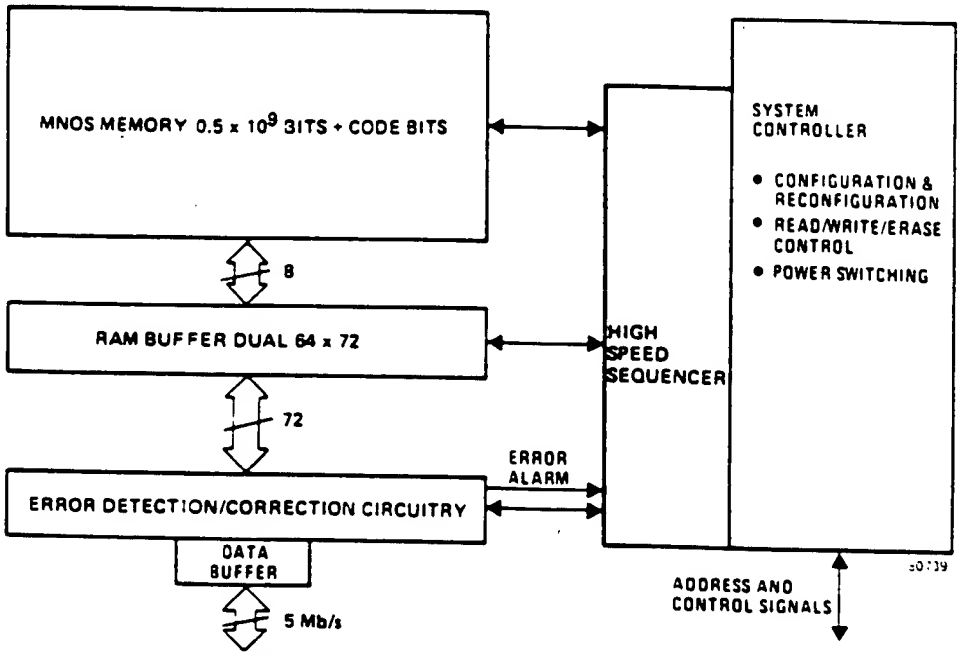


Fig. 7. Memory system organization of a gigabit mass memory.

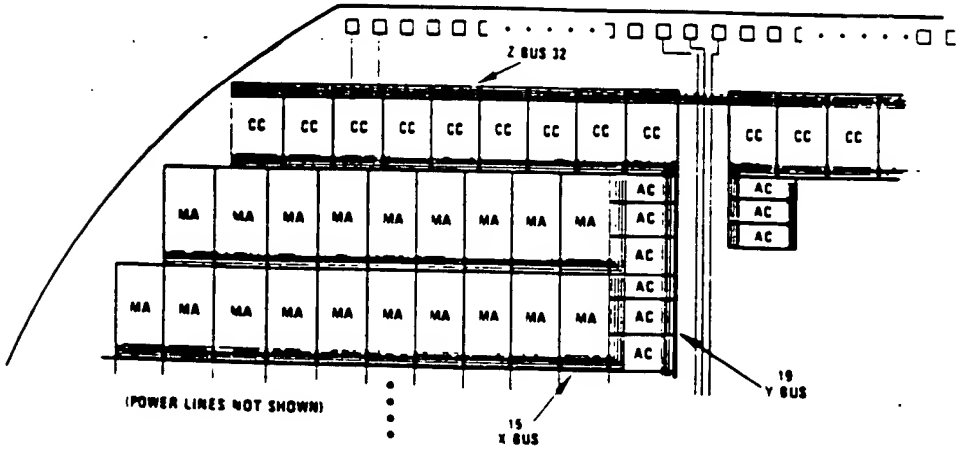


Fig. 8. Wafer memory architecture.

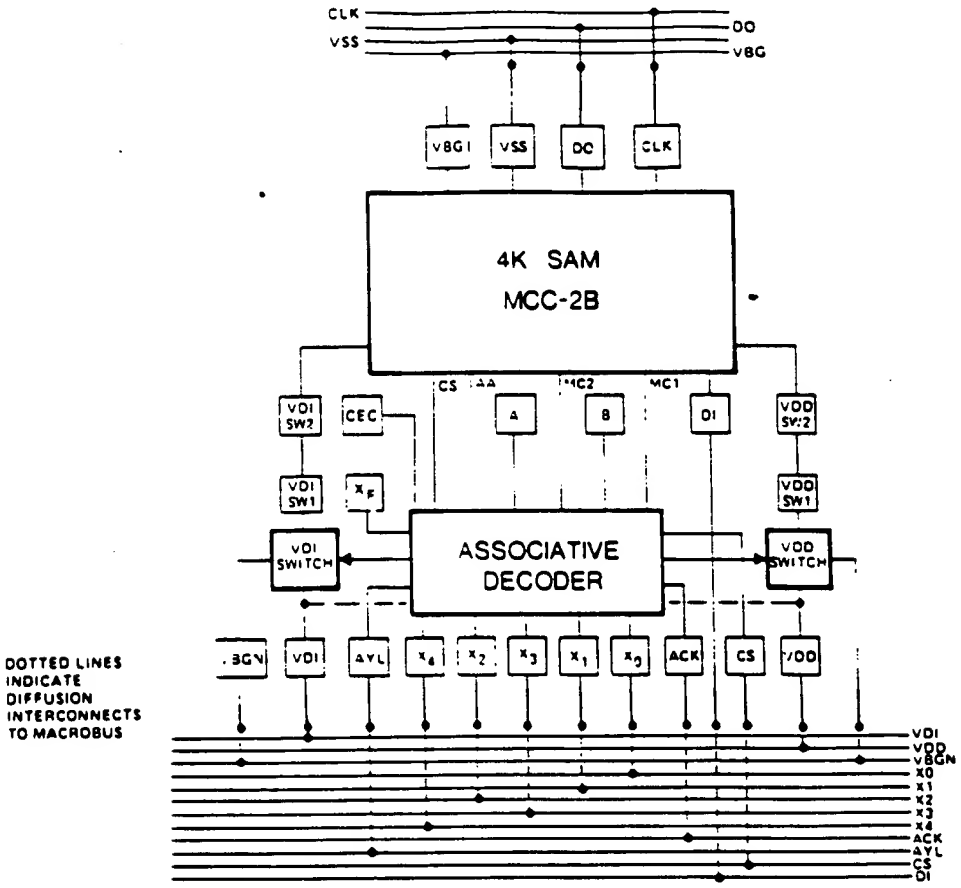


Fig. 9. MAW-1 overview: associative decoder, SAM with internal pads, and macro-bus.

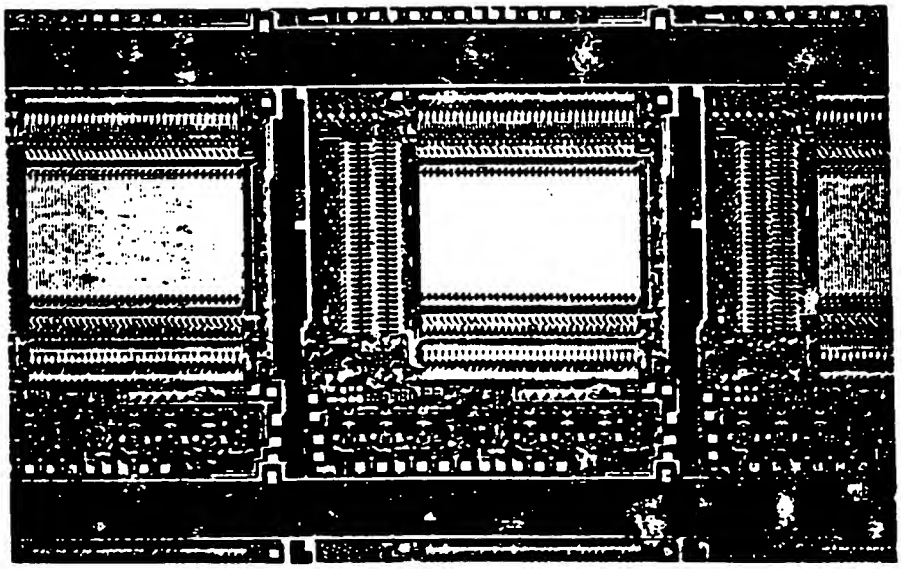


Fig. 10. Photomicrograph of MAW-1, the wafer-level interconnected 4096 bit sequential access memory arrays.

LEX14277

correction circuitry also performs a bit transformation on the encoded data to ensure that each bit of the data word is stored in an independent memory array.

The channel and array controllers resident in the memory wafers implement the memory addressing technique, by selectively turning on power, switching data and control information to the intended group of memory arrays. This addressing structure minimizes the power consumption of the memory wafers. The memory wafers constitute the bulk of the memory system and contain many of the unique features of the system.

§4. The Reconfigurable Memory Wafer

As a first step towards the implementation of a memory wafer for mass memory application, a reconfigurable memory wafer (MAW-1) was developed and studied. The MAW-1 is composed of interconnected memory arrays for the purpose of evaluating the circuits necessary to accomplish power switching, address storage and decode address control for interconnect reconfiguration. Figure 9 is an overview of the MAW-1 which shows the physical arrangement of the device. Figure 10 is a photomicrograph of the MAW-1 circuit array illustrating also the wafer level interconnect, or the macrobus.

A 4096 bit nonvolatile sequential access memory (SAM) was the basic memory storage array. Several features of the SAM array uniquely establish its application in an AWSI memory: (1) Memory storage is nonvolatile, based on n channel MNOS storage. (2) Data input is high impedance and buffered with an input latch; data output is tri-stage, and buffered for bus driving. (3) To reduce interconnect bus lines, access within the SAM storage array is sequenced internally, timing input is restricted to a single clock line, and operation instructions are decoded internally. (4) For multiplexed operation in a bus interconnected system, a chip select control and a flag bit are provided. Table I summarizes the SAM device characteristics.

Table I. SAM characteristics

• Technology	Metal gate MNOS/NMOS, 6 micron process
• Circuit type	Static logic, quasi-static shift register, single-element-per-bit nonvolatile storage
• Organization	Sequential access, organized 64 words × 64 bits
• Cell size	$37 \times 20\ \mu m^2$
• Die size	$167 \times 144\ mil^2$
• I/O data rate	1.0 MHz
• Block erase	1.0 msec
• Write/word	200 μsec
• Data transfer/word	3.0 μsec
• Read/word	3.0 μsec
• Supplies	+20 V, −10 V, GND
• Power	<250 MW
• I/O pinout	12

The mechanization of on-the-wafer interconnect is accomplished with the development of an associative interconnect. The associative interconnect consists of six major components: (1) The macrobus which is the system bus, (2) power switches, (3) power connect/disconnect logic, (4) nonvolatile address memory, (5) address memory control, (6) address comparators.

The associative interconnect was tested for (1) address memory storage and pattern insensitivity, (2) power-on control to the SAM by associative decoder address match and mismatch, (3) power turn-off of the SAM by address match, (4) permanent programmable match disable to eliminate bad decoder circuits and/or SAM's, (5) power response time of 5 μsec. With the associative interconnected tested for operation, the MAW-1 was tested for accomplishment of (1) selective power switching via the macrobus, (2) selective control of SAM circuit operation (read, write and erase) from the macrobus, and (3) lockout of defective SAM's. The successful completion of the MAW-1 tests led to the design of a megabit wafer memory for a prototype mass memory.

§5. Fault Tolerant Systems

Other than memory applications, the unique architectural structures available via AWSI are applicable to general purpose fault-tolerant computer systems. Some of these structures are: on-wafer spares, nonvolatile memory switches to maintain on-wafer configuration, associative address decoders to provide hard and soft addressing capabilities, nonvolatile memory configuration and status tables to preserve the state of a system during power interrupts and failures, and small highly testable components which are combined to form a large, on-wafer fault-tolerant system.

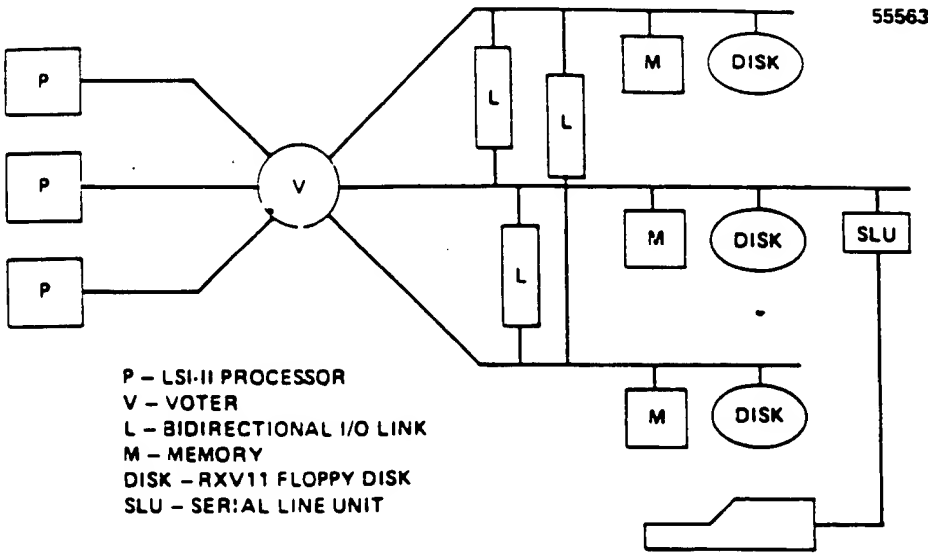


Fig. 11. C. vmp bus level configuration.

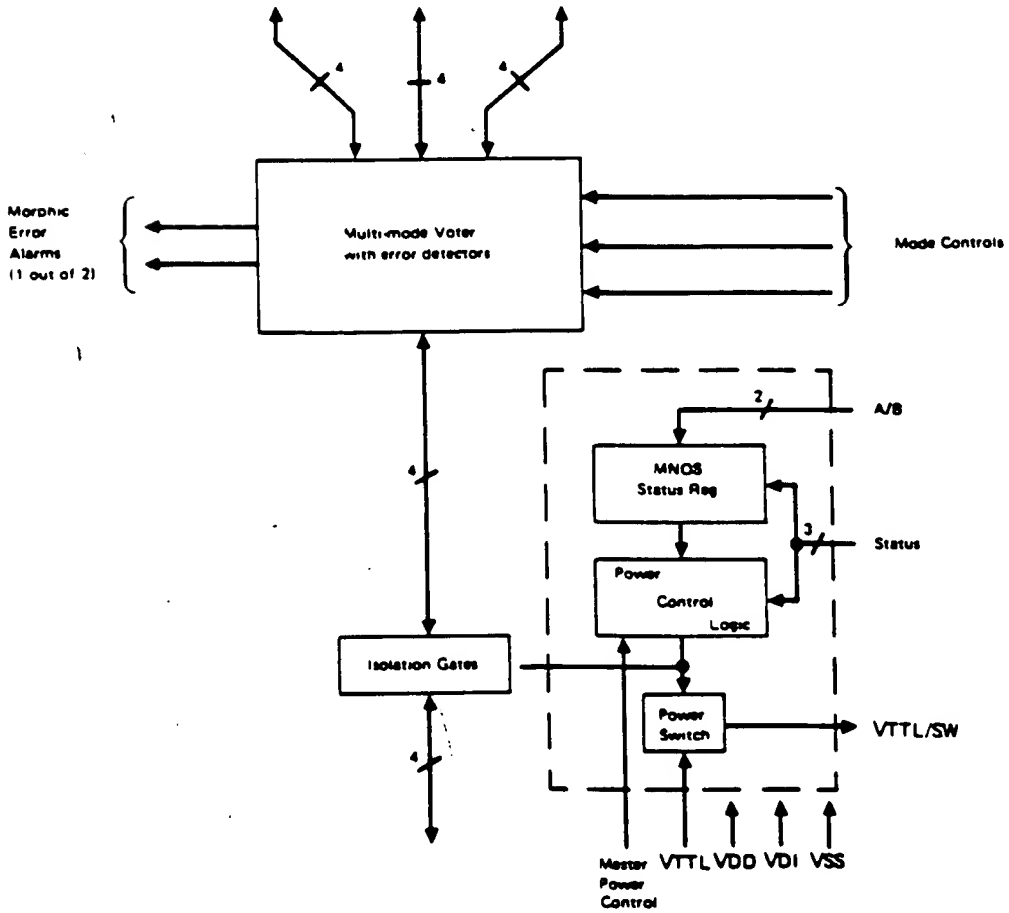


Fig. 12. Adaptive C. vmp voter circuit showing MNOS status register and power switch.

LEX14279

Let us consider an adaptive C.vmp voter circuit for application in a fault tolerant system. The C.vmp is developed at Carnegie-Mellon University, and stands for Computer (voted multiprocessor).[12,13] It was designed specifically as a machine for the study of fault tolerance and the effects of faults, both transient and permanent, on the system. C.vmp, shown in Fig. 11, uses three DEC LSI-11 computers organized in a TMR configuration. The majority voting of the TMR system occurs at the interface of the processors to the system buses so the voter is between the processors, and the memories and peripherals. This allows voting to be done every bus cycle. Program studies have shown that every memory element of the system is voted upon regularly, except the stack pointer, without any special software.

Two of the major design considerations were that fault tolerance be carried out in real time, and that it be transparent to the software; this dictated a TMR configuration. Another design consideration was that the system user be able to dynamically trade fault tolerance for throughput. This characteristic resulted in a system which can operate in three modes: a loosely-coupled three computer mode, a fault-tolerant TMR mode, and the broadcast mode which allows a nonredundant peripheral to communicate with all three of the processors. Each of these is controlled by the state of the voter.

With the use of MNOS nonvolatile storage, an adaptive C.vmp voter is implemented, Fig. 12. The flexibility of the adaptive C.vmp voter, is apparent from the wide variety of voting networks which can be constructed with different arrangements of the voter. The voters can be cascaded together to form either a unidirectional or bidirectional voting network of

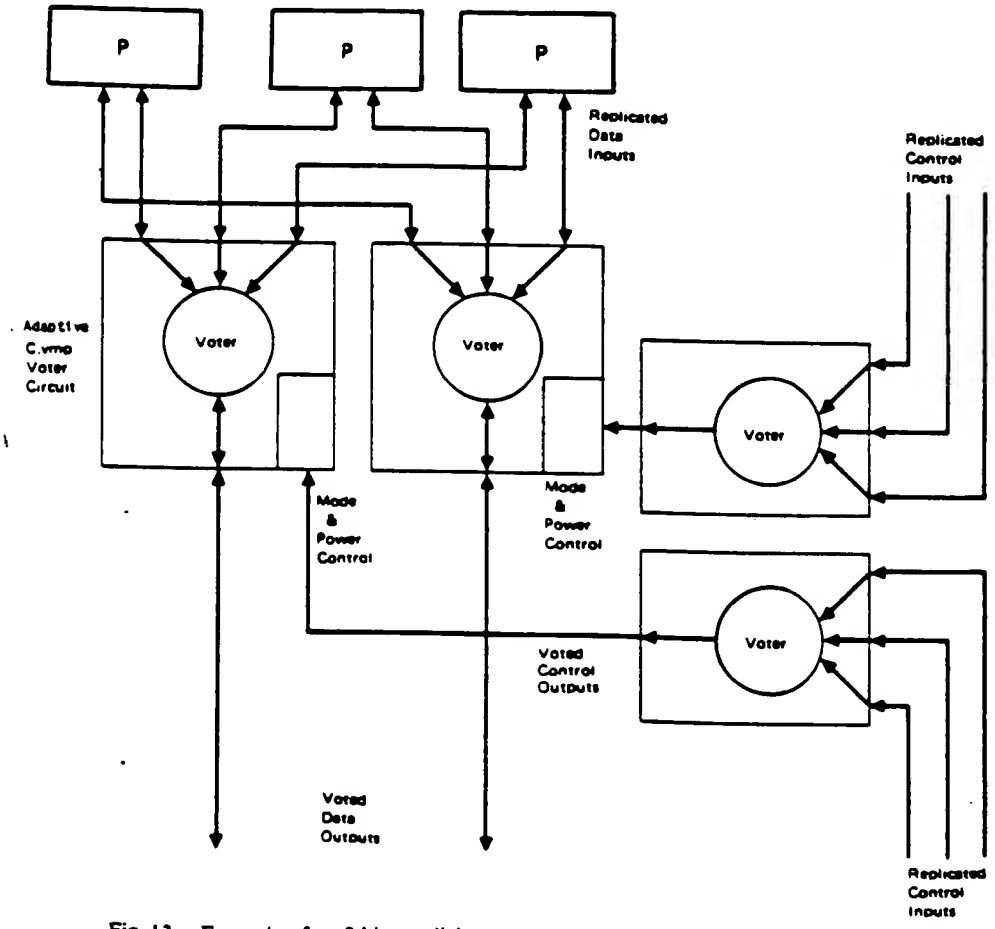


Fig. 13. Example of an 8-bit parallel voting data path with voted control signals.

LEX14280

any width. (Here unidirectional and bidirectional strictly apply to the voting, the data path itself is always bidirectional through the voting circuit.) An 8-bit unidirectional voting network is illustrated in Fig. 13, where two 4-bit voters supply the 8-bit voted data path and two additional voters provide the voted control inputs. A bidirectional 8-bit voting network would simply consist of two unidirectional networks connected together, with one being the mirror image of the other. The voting on the control inputs can also be configured to the desired application. Each voter's power switch and mode control inputs can be connected to a single set of voting inputs, a multiple set of separate voting inputs, or any combination. The adaptive voter circuitry, can be used to implement the hybrid redundancy through the use of MNOS controlled power switches. As illustrated in Fig. 13, whenever one of the microprocessor fails, its power is removed and a spare unit is powered up to replace it. The nonvolatile MNOS status memory retains the proper configuration of functional microprocessors even after a system power down. Should enough of the microprocessors fail so that no spares remain, the voter circuit is switched to multiprocessor mode, so that the system operates with a single microprocessor plus a spare thus circumventing the coverage probelm of TMR systems when catastrophic failure resulting in only two operating units out of three units necessary for majority voting.

§6. Concluding Remarks

Based on a nonvolatile semiconductor memory device, the MNOS transistor, an interconnect technology is developed. The application of the interconnect technology, in connection with the memory transistor, results in electronic systems which can be characterized as highly reliable and fault tolerant. Further development of the MNOS based adaptive circuits and interconnect undoubtedly will lead to self-organizing machines with artificial intelligence.

References

1) Y. Hsia, G. C. C. Chang: *Proc. IEEE 1979 National Aerospace and Electronics Conf.* May (1979) 881.
2) R. L. Petritz: IEEE J. Solid State Circuits SC-2(1967) 130.
3) D. F. Calhoun: AFIPS Conf. Proc. 35(1969) 99.
4) J. C. Hunter: *Symposium on Advanced Memory Concepts, Stanford Research INSTITUTE, Melon Park, CA* (1976) 450.
5) R. C. Abusson and I. Catt: IEEE J. Solid State Circuits SC-13(1978) 339.
6) Y. Egawa, N. Tsuda, K. Masuda: Digest of Tech Papers IEEE Int. Solid-State Circuits Conf. (1979) 18.
7) J. J. Chang: Proc. IEEE 64(1976) 1039.
8) M. H. White and J. R. Cricchi: IEEE Trans. Electron DevicesED-19(1972) 1280.
9) Y. Hsia: IEEE Trans. Electron DevicesED-25(1978) 1071.
10) Y. Hsia: IEEE Trans. Electron Devices ED-24(1977) 568.
11) M. White, J. W. Dzimianski, M. C. Peckerar: IEEE Trans. Electron Devices ED-24(1977) 577.
12) D. P. Siewiorek, V. Kini, H. Mashburn, S. McConnel, M. Tsao: Proc. IEEE 66(1978) 1178.
13) D. P. Siewiorek, V. Kini, R. Joobbani, H. Bellis: Proc. IEEE 66(1978) 1200.

ADAPTIVE INTEGRATION TECHNOLOGY DEVELOPMENT

VOLUME III
MASS MEMORY SYSTEM DESCRIPTION

PART 1

23 MAY 1979

ADAPTIVE INTEGRATION TECHNOLOGY DEVELOPMENT

VOLUME III
MASS MEMORY SYSTEM DESCRIPTION

PART 1

23 MAY 1979

Prepared by _____
W. Weia
Project Engineer

Approved by _____
W. A. Geideman
Chief Engineer

Approved by _____
R. R. Erkeneff
Program Manager

LEX15409

Volume III

Forward

The final report is contained in eight separately-bound volumes as follows:

Volume	I	Final Report Adaptive Integration Technology Development
Volume	II	Technical Progress and Accomplishments
Volume	**III**	**Mass Memory System Description - Part 1**
Volume	IV	Mass Memory System Description - Part 2
Volume	V	Reference Data and Analysis
Volume	VI	Error Control for a Mass Memory System
Volume	VII	Reduced Geometry Process Development Effort
Volume	VIII	Wafer Material Log

The boxed volume indicates the one contained in this binding.

Volume III
TABLE OF CONTENTS

Volume III
LIST OF ILLUSTRATIONS

LEX15414

LIST OF ILLUSTRATIONS (CONTINUED)

LIST OF ILLUSTRATIONS (CONTINUED)

Volume III
LIST OF TABLES

Section 1
MASS MEMORY SYSTEM DESCRIPTION (PART 1)

1.0 INTRODUCTION

The next generation mass memory system should and can possess all of the following characteristics:

- Low power consumption
- System level nonvolatility
- High data rate (equivalent to computer main memory)
- High reliability
- Ability to reconfigure and/or gracefully degrade
- Fault tolerance
- Functional modularity and expandability
- Low weight
- Small volume
- Low cost

This report describes a mass memory system that will meet all of these demands. Initial efforts have defined the technology that will support the development of a light-weight, small volume, low power, 0.5×10^9-bit semiconductor memory. One application could be a highly-reliable replacement for presently employed magnetic tape recorders.

Two highly innovative semiconductor technologies are proposed to achieve this goal; adaptive wafer scale integration and an n-channel metal-nitride-oxide-semiconductor (MNOS) transistor memory device. These will satisfy the immediate objective to develop a cost-effective functional substitute for the magnetic tape recorder. One or both of these technologies can be the basis not only for memory systems, but also for computers or encryption equipment. They are generally applicable to any complex information processing and storage requirement where cost, reliability, size, and/or weight are important.

1.1 ADAPTIVE WAFER SCALE INTEGRATION (AWSI)

High system reliability, computed to be greater than a 0.95 probability of system operation over 5 years, is obtained by using a new and unique integrated circuit technology called adaptive wafer scale integration (AWSI). It permits large numbers of memory arrays to be interconnected (repeatedly) by electrically alterable, nonvolatile, latching interconnect circuits. Both memory arrays and latching interconnect circuits are fabricated on the same semiconductor wafer substrate through a series of photolithographic process steps. The latching interconnect circuits provide control for address, power and data bus lines fabricated by metallization steps. The use of these interconnect circuits results in a minimization of bonding pads and in a reduction of bus line loading which results in lower power requirement. Because these interconnect circuits are made as a part of the semiconductor wafer fabrication process, the connection of logic function and memory circuit arrays to the buses on the wafer eliminates large numbers of potentially unreliable electrical connection bonds, often the most unreliable part of an LSI circuit, plus substantial amounts of manufacturing labor and material.

Under the control of peripheral error detection/correction and built-in test circuits, the interconnect circuits enable malfunctioning arrays to be replaced automatically by spare arrays, which are available on the wafer. Since the interconnects are electrically alterable, this sparing process can be carried out repeatedly when required. Because they are nonvolatile, the interconnect configuration is retained over periods of power down or power interruption.

The basic interconnect circuit, which has been developed in an independent research and development program, is a standard logic or buffer gate NMOS transistor circuit that is enabled by an electrically-alterable, nonvolatile, MNOS memory cell. The use of this circuit to interconnect a shift register to a data bus is shown in figure 1-1. This particular example circuit has been fabricated and integrated on a wafer where a shift register in a given horizontal string may be either connected in series with the data line or bypassed (to simulate the removal of a faulty register from the bus).

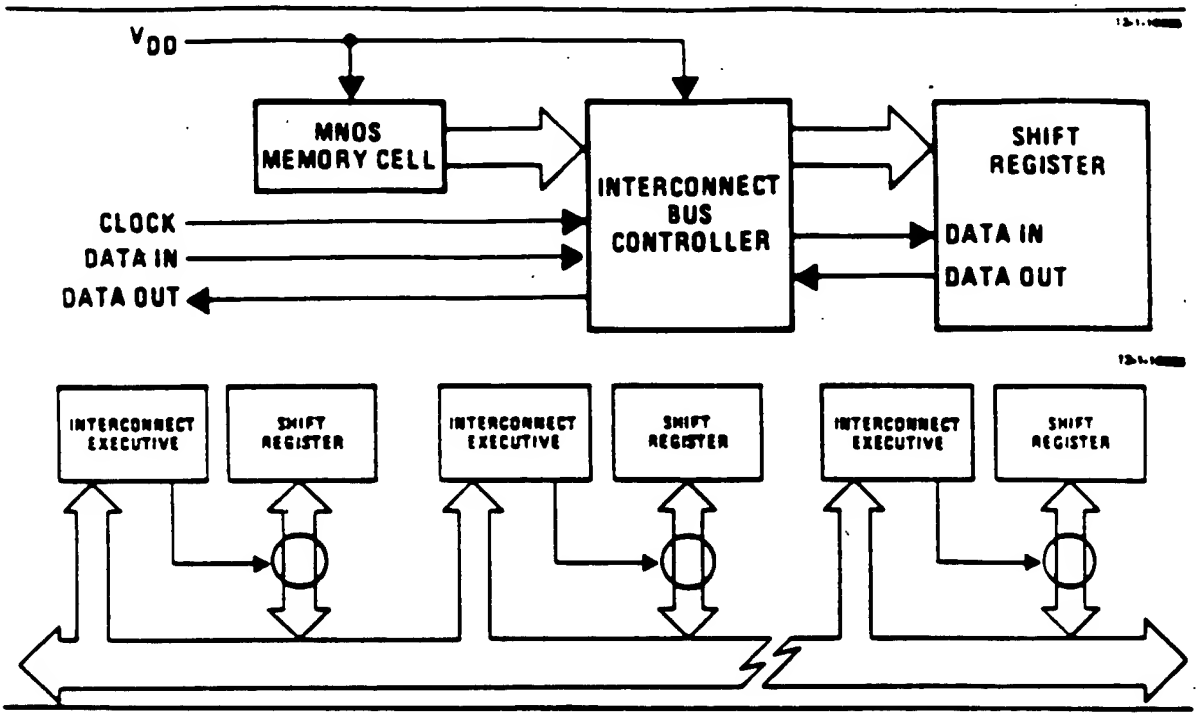


Figure 1-1. Use of AWSI to Interconnect a Shift Register String

The mass memory system, described in detail in following sections, combines the AWSI interconnect technology with a nonvolatile n-channel MNOS transistor memory device.

On-the-wafer interconnect is mechanized such that small sequential access memory arrays with a memory storage capacity in the range of 4k to 8k bits per memory array are interconnected to obtain a large capacity store approximately 0.5×10^6 bits in a single wafer.

1.2 PRELIMINARY SYSTEM SPECIFICATION (DESIGN GOALS)

The preliminary specifications for a 0.5×10^9-bit mass memory system which are presented in table 1-1, are for a sequential access mass memory functionally intended as a magnetic tape recorder replacement. The memory system design philosophy is governed by the following major factors. The storage level of

TABLE 1-1. PRELIMINARY SPECIFICATION FOR 10^9-BIT AWSI MEMORY

TECHNOLOGY

SYSTEM

Nonvolatile self-repairing, electrically-alterable interconnect and unpowered redundant sparing.

COMPONENT

Nonvolatile memory (NVM)/sequential access memory arrays-using wafer scale integration.

PROCESS

MNOS/NMOS
3-inch wafer (initial development)
4-inch wafer (engineering model)

STORAGE CAPACITY

(Wafer level) 0.5 Mbits (4-inch wafers)
(System level) 0.5 x 10^9 data bits

FEATURES

Nonvolatile storage
Self-repairing
File organized (256 Kbits/file; 2048 files total)
Power switching
Error detection/correction

I/O SERIAL DATA RATE

5 Mb/s

POWER-ON TIME

1 s exclusive of power supply

FILE ACCESS TIME

1 ms random access

POWER

	Peak Power	Average Power	Total Power Consumed
READ	27.5 W	25.7 W	3.8×10^{-4} Whr/file
ERASE	37.0 W	33.4 W	5.9×10^{-5} Whr/file
WRITE	38.4 W	35.9 W	5.3×10^{-4} Whr/file

RELIABILITY

5 year reliability of 0.95

VOLUME

2,200 cu inches

WEIGHT

60 pounds

0.5×10^9-bits and the data rate of 5 Mb/s are selected as typical numbers for a recorder application. The system reliability is specified as greater than 0.95 for five years of unattended operation; the system design employs error detection and correction circuitry and unpowered spare memory elements in order to perform in-situ repair of the memory system to maintain full capability for the five-year period. This technique for high reliability is extended beyond the memory arrays themselves to provide built-in test and sparing for other parts of the mass memory system which include the system controller and the basic error detection circuitry.

The memory capacity of 0.5×10^9 bits of data, with a 12% growth for error detection and correction coding, results in a total storage requirement of approximately 0.6×10^9 bits of information. The total system reliability can be achieved with a 5 percent sparing level. Thus, the total memory must include storage capacity for data, code bits, and spare memory elements. The file size of 256 kbits is chosen as a representative memory requirement. The basic element of the mass memory system is the memory wafer, which is organized in four separately operated quadrants. No restrictions are placed on similarity of each quadrant in order to provide the most versatile system design and to minimize the cost of fabrication.

The requirement for minimization of both average and peak power results in several compromises in the system design, most notably a limitation on the speed of the system. However, the overall speed of the system can be increased with relaxation of the power requirements. Particular attention is given to power sequencing of the components to minimize total power consumption. Much of the system overhead is allocated to this power sequencing.

The memory system organization assumes that the memory is written into in one time interval and read in another. No provision is made in the current system to read one section of the memory while writing in another section. However, modifications to the system controller can be made to allow such operation. In this respect, the operation is similar to that of a conventional tape recorder.

The error-correction technique is predicated on the detection of errors during operation and then performing built-in test and reconfiguration off-line, i.e., when the basic memory system is not in use.

1.3 SUMMARY

This report presents the results of the system design activity. Section 2.0 is the system hardware architecture and includes brief descriptions of the mass memory system components, including the wafer and its circuit elements and the controllers external to the basic wafer organization. The hardware description of section 3.0 presents the details of the mass memory system components. Section 4.0 presents the details of the memory system operation, concentrating on the addressing procedures and the memory wafer command sequencing. The system software of section 5.0 presents further details of memory operation, including the procedures for built-in-test and reconfiguration. Section 6.0 describes the procedures to be used in testing the memory wafers, including the wafer acceptability criteria. The yield of the memory wafers is described in section 7.0. Section 8.0 discusses the reliability of the memory system, and section 9.0 details the memory system packaging, including the memory wafer packaging. Sections 1 through 5 are presented in this volume. Sections 6 through 9 are presented in Volume IV, Mass Memory System Description (Part 2).

Section 2
SYSTEM HARDWARE ARCHITECTURE

2.0 AWSI MASS MEMORY ARCHITECTURE

The design approach for the mass memory system has emphasized the minimization of system power consumption while meeting the operational specifications described previously. The major factors impacting the system design include the minimum addressable file size, the minimum bit error rate in the data, and the requirement to provide recovery of the system after failure of any single element. The total capacity of the memory has been specified as 0.5×10^9 data bits. The memory elements have been organized into wafer quadrants in order to provide maximum versatility and to allow the maximum number of the processed wafers to be employed in the system. The organization does not require a fixed minimum memory capacity per wafer or per quadrant, but rather the system is organized to accommodate varying capacities on the wafers which make up the memory system.

The reconfigurability of the memory to achieve the 0.95 reliability for 5 years will be achieved via single-bit error correction/double-bit error detection, on-line reconfiguration, and built-in test (BIT) programs. Replacement of failed circuits will be accomplished when the memory is not being actively addressed.

Utilization of the functional circuits while avoiding the malfunctional ones has to be accomplished in such a manner that the overhead circuits constitute a reasonable fraction of the total in order that the desirable memory characteristics are effectively preserved and the resulting system remains manageable. Two main techniques are employed: a partyline bus concept where units on the bus respond upon recognition of their unique code (for that particular bus); and on-wafer mapping where an off-wafer controller need not be concerned as to which circuits are functional or malfunctional during data I/O modes.

The on-wafer control circuits are designed to take maximum advantage of and be consistent with the process used in the memory array fabrication. The implications of this are slow circuit operation with no high-speed clocks, a minimum amount of sequential logic, a liberal usage of nonvolatile memory storage (highly structured), and a bus-oriented design. At all junctions throughout the system design the simplest possible alternatives are implemented consistent with system requirements. This simplicity results in low overhead, a highly structured system, and a minimum of special considerations.

2.1 SYSTEM ORGANIZATION

The total system is composed of four major components; three are in the data path, the other is the system controller, as shown in figure 2-1. The MNOS data storage contains 0.5×10^9 bits of data plus 0.06×10^9 bits for the error detection/correction code. Thus, 12.5% added capacity is required for the error

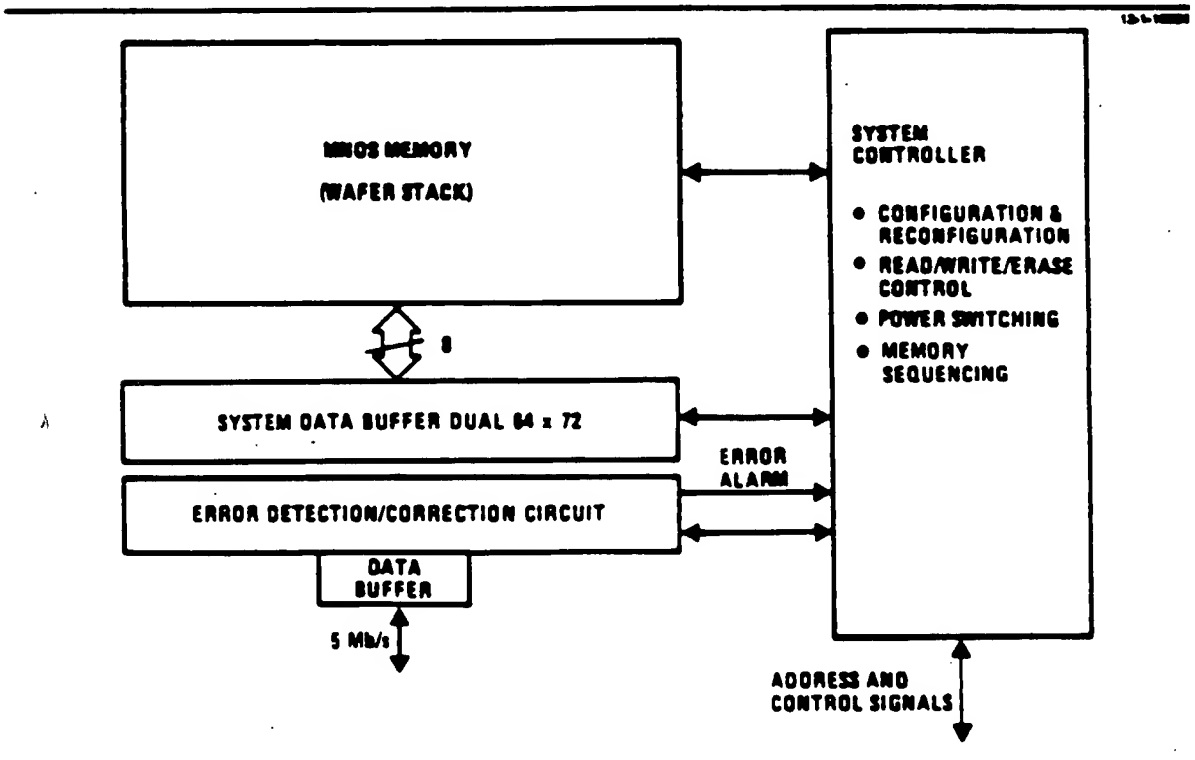


Figure 2-1. Memory System Organization

correction function. The MNOS memory is organized into a wafer stack which contains storage for the code bits, spare functional memory, and the data memory. Each wafer is divided into logically independent quadrants containing many 4096-bit memory arrays. A wafer is usable if it meets a predetermined requirement for the total number of operable memory arrays. No particular spatial distribution of usable arrays within a quadrant is required.

A system data buffer memory interfaces between the MNOS memory and the error detection/correction circuits. This dual 64-word by 72-bit buffer memory functions as a data bit transposition unit. When data enters from the error detection/correction circuits, it is stored 72 bits to a word (64 data bits plus 8 code bits),until the 64-word system data buffer is filled. The filling of an alternate data buffer memory is then begun. Meanwhile, the first buffer memory is unloaded one bit position at a time, i.e., bit position 0 for all 64 words, then bit position 1, then 2, etc. This bit transposition ensures that each bit of the 72-bit encoded word is stored in a different memory array. This bit-slice architecture ensures that should any single memory array fail, the lost data may be reconstructed by the error correction circuitry.

The error detection/correction circuitry appends an eight-bit code onto every 64 bits of incoming data to produce a 72-bit word. (This word length has no relation to external word lengths since incoming data is serial and there are no word markers.) The error detection/correction code has sufficient capability to correct any single error as well as to detect many multiple errors. Each bit of this 72-bit word is stored in a different 4k array. Thus, it is most likely that any memory failure error will be corrected and hence masked, as data are transferred out of the system. The user will be unaware of the internal failure; his corrected data will be received at normal speed. A faulty memory array will be replaced at a later time by the reconfiguration routine. A comprehensive description of this routine and the method of error detection/correction coding is given in the report, "Error Control for a Mass Memory System," which is included as a separate volume.

The system controller coordinates the timing between the various units in the data path. It provides built-in-test (BIT) capability as well as reconfiguration control for the following:

- For MNOS wafer components: replacing faulty memory arrays and controller circuits with spares on the wafer; also switching out entire faulty quadrants and, where necessary, wafers.

- For system data buffer memories: switching in spare memory chips.

- For error detection/correction circuits: switching in spare logic circuits.

The switching in of spare units is facilitated by extensive use of tri-state buses, power control, and BIT between data file transfers. The system controller itself has spare modules that may be switched in when malfunction is detected during BIT.

It is thus seen that the redundancy built into the system provides high reliability for extended missions.

2.2 MEMORY WAFER ORGANIZATION

An architecture is needed that can make maximum use of functional memory arrays dispersed among malfunctional arrays without using a disproportionate amount of silicon area for overhead functions. Figure 2-2 presents an overview of such an architecture. A wafer is partitioned into four quadrants; each quadrant possesses an independent operational capability. Within a quadrant, memory arrays (MA's) are arranged by rows, with each array in the row connected to a common row bus (X-bus). There are no unique lines to a memory array for chip select as is commonly utilized with memories organized with discretely packaged units. Rather, address selection is accomplished over a 4-bit bus activated by an array controller (AC) located at the interior end of the row. Defective memory arrays are simply not accessed by the AC. With most defects in a memory

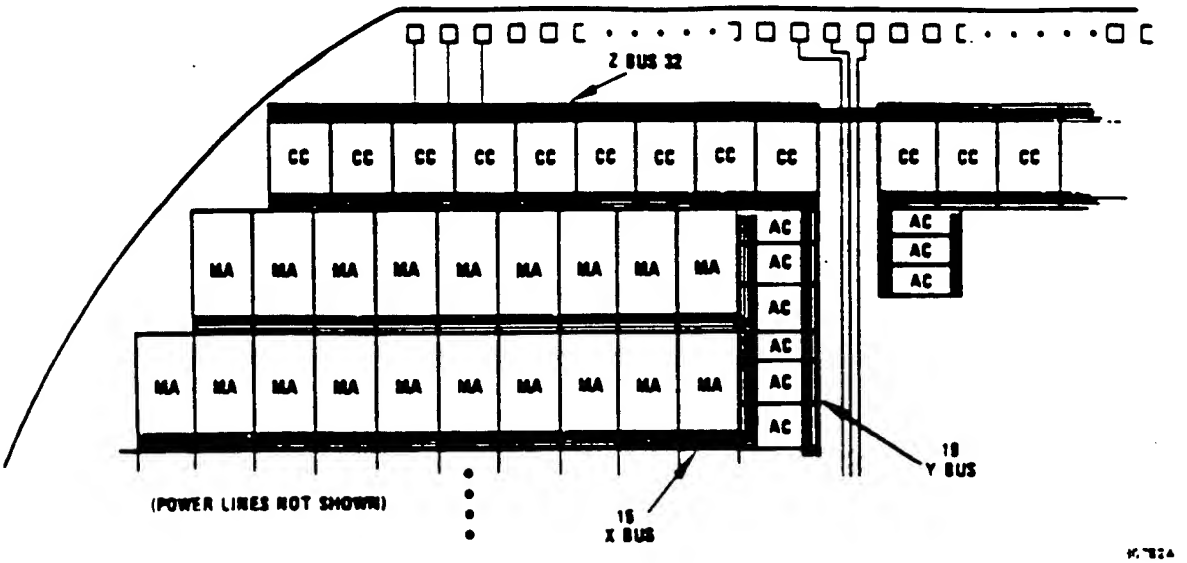


Figure 2-2. Wafer Layout

array, no other consequences are assumed manifest. However, in the less probable cases where a defective array makes its neighbors inaccessible or inoperable, the entire row is abandoned by never accessing any memory array within that row and thus never using the AC controlling that row. Two different types of such faults may occur. In one type, an array would draw excessive power; in another, the array would make the X-bus inaccessible. To rectify the first fault, since the power is switched at the AC, it can be turned off to that row where the faulty MA resides, while power is still provided to other rows. A similar operation is available for the bus lines at the AC. It can be noted from figure 2-2 that all the bus lines for a row are shown connected to a unit of three AC's. The entire X-bus is buffered at this point. Thus, X-bus faults are not propogated throughout the quadrant but rather are confined to a single row.

The AC provides two main functions: one is gating between the row and quadrant, as described above; the other is routing data and control information

between the channel controllers (CC's), which are located in the first row on the quadrant, and the memory arrays. The routing function is effected in two separate steps. First in sequence, the AC is addressed and locked onto the Y-bus which carries information between a CC and AC's. Once an AC is locked onto a bus, it responds to all commands on the bus until unlocked. The second sequential step is to translate memory array numbers into actual memory array addresses, to present the memory address on the X-bus to lock the addressed memory on the X-bus, and then to transfer information between the Y and X buses.

Each row is provided with three AC's: one must be operational to make the row accessible. The additional functional AC's (up to two) are used as spares in the event of failure of the active AC.

The quadrant's first row contains several CC's; only one must be operational to make the quadrant usable. However, most quadrants should have at least two operable CC's per quadrant to ensure reliability. The major functions of a CC are to:

- Recognize its quadrant address and lock onto the Z-bus which carries information to/from the system controller outside the wafer.
- Translate data block numbers into their AC addresses.
- Buffer data and power lines between the Z-bus and the Y-bus.
- Route data and commands to/from the AC.

There are, therefore, two distinct levels of control in the wafer architecture. The CC interfaces with the off-wafer controls, responds to selected commands and routes commands and data, where intended, to the AC. The AC interfaces with the CC, responds to selected commands and, in turn, routes commands and data to the MA which responds to commands and reads/writes data.

The bulk of a quadrant is composed of a number of memory arrays (MA's). The MA's utilize nonvolatile MNOS memory transistors, which allow data retention without any standby power. Each MA can store 4096 bits of data. This data is accesed serially in groups of 64 bits, at a 2 Mbit/sec rate. Four MA's are active concurrently in a single quadrant, i.e., 256 bits of data are accessed during any read/write cycle of a quadrant.

The four MA's are accessed by first addressing the CC of the proper quadrant. The AC's controlling two separate rows of MA's are then addressed, followed by a pair of MA's in each row. Once addressed, these four MA's act as a single unit with the CC and AC's transparent to the data flow.

Two such quadrants are operated in parallel to allow the access of 64 8-bit words, or 512 bits, every read/write cycle of the memory. This dual quadrant configuration is used to obtain the desired 5 Mbit/sec system data rate. The number of quadrants operating in parallel can be increased to give higher data rates. Simultaneous control of the dual quadrants is performed by the system controller.

2.3 SYSTEM CONTROLLER

The off-wafer subsystem, including the system controller is shown in figure 2-3. The system controller is functionally and geometrically partitioned into three sections; each section has independent timing. They are:

- Power kernel
- System microprocessor
- High speed sequencer.

All sections are replicated and power switchable to achieve specified reliability requirements. The power kernel switches power to operational sections of the system microprocessor CPU and to itself. This self-switching is accomplished by majority voting at each of several redundant identical processors of the power kernel. Initially, power is supplied continuously to all operational power kernel processors. As units malfunction, they are powered down, and once

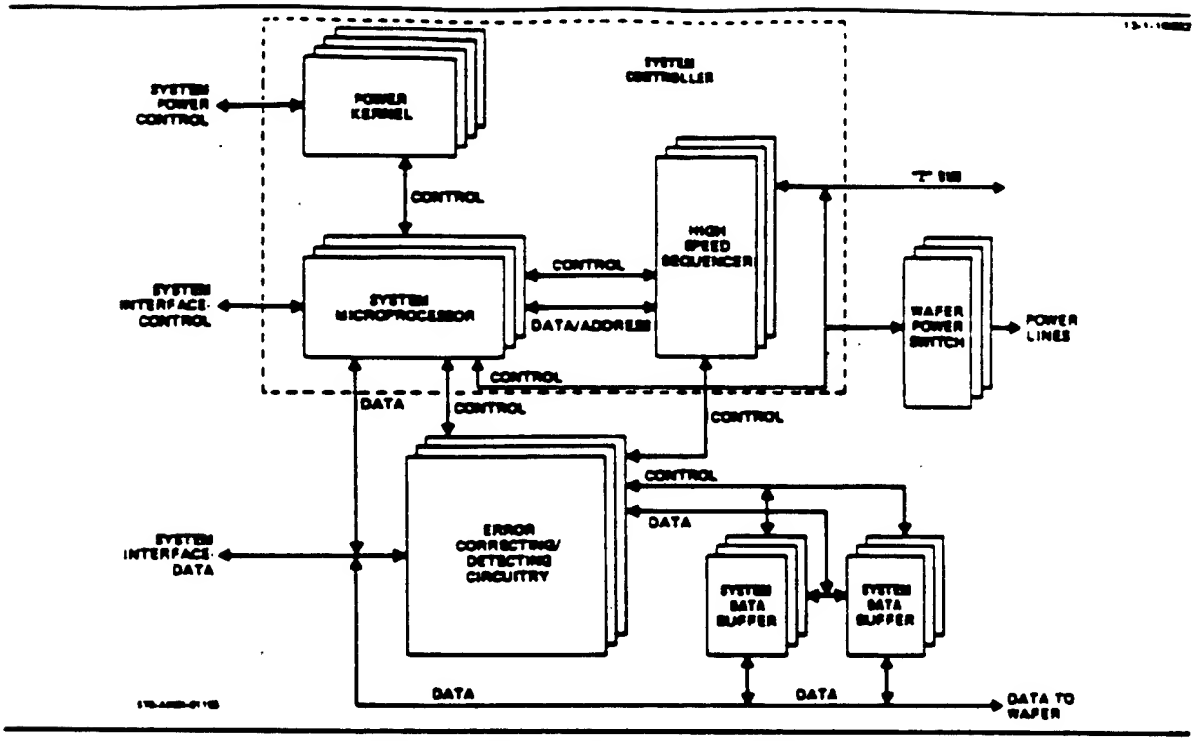


Figure 2-3. Off-Wafer Subsystem

powered down power can be turned on again only through remote control. Spare units are powered up as replacements, so that a full complement is operational until such time that all spares are exhausted. This technique ensures operational power control as long as at least two power kernel processors out of the original complement are operational and the units fail one at a time.

Operationally, a power kernel processor has responsibility for comparing the outputs of its replicates, checking the health of the system microprocessor and power switching of itself and the system microprocessor. The power kernel is a relatively simple single-chip microprocessor.

The system microprocessor controls the data operations for read, write, and erase; responds to system commands; configures and reconfigures the system except for its own bootstrap kernel and the power kernel; carries out diagnostics and BIT; and switches power to the high-speed sequencer, the memory

components (wafers, quadrants, CC's, AC's, MA's), the system data buffer memory and the error detection/correction circuitry.

The system microprocessor is a fairly sophisticated unit with relatively high speed capability. It must be contained in a few LSI circuits to make power switching of its sections practical. The new 16-bit high speed microprocessors meet these requirements. Reliability considerations will dictate the level of replication needed to satisfy the specified requirement.

The high speed sequencer (HSS) provides the detailed timing and sequencing to execute the commands from the system microprocessor. Although the HSS is relatively simple, it is a very high speed unit. It is composed of sequence control chips, ROM's, and related chips. For the specified reliability, the level of replication will be determined.

By partitioning the memory-system peripheral circuits to a relatively small replaceable unit level, high reliability can be achieved with a minimum of back-up hardware and with no power loss to "hot" standby circuits. The operational characteristic that allows the use of "cold" standbys is "repair between data file transfers." Combined with real time single-data-error correction, this technique provides the desired system reliability with reasonable expenditure of power and weight.

2.4 DATA FLOW

Incoming and outgoing data rate capability ranges from dc to 5 Mb/s bit serial rate as shown in the data flow chart of figure 2-4. For a data file of 256 kb, read access is less than 1 ms; write access is approximately 7 ms (presuming an erase cycle before write).

Data transfer out of the system data buffer memory is at an 8-bit word rate of dc to 2 Mb/s peak or bit rate to 16 Mb/s. The MNOS memory accepts data at the same dc to 2 Mb/s peak word rate for a 8-bit word, or 16 Mb/s bit rate.

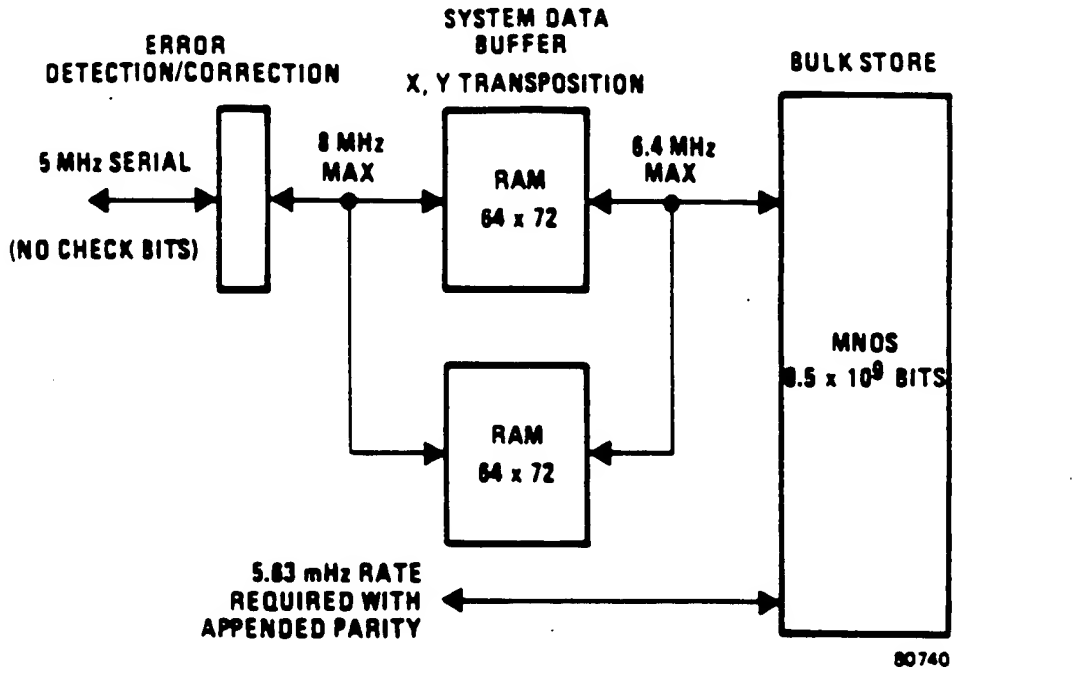


Figure 2-4. System Data Flow

The 5 Mb/s serial data rate at the user interface becomes 5.63 Mb/s after the 8 error bits are appended to the 64 data bits. The difference between the capability (16 Mb/s) and the requirement (5.63 Mv/s) provides time for overhead functions. At the required rate, approximately 52 ms are required to load/unload a 4096 x 72-bit file. Additionally, approximately 7-ms erase time precedes a data file write. Two CMOS/SOS ping-ponged system data buffers are sufficient for the designed data rates. Thus, no supporting system component limits the data transfer rate.

2.5 WAFER POWER MANAGEMENT

To minimize power consumption, power switching is implemented at various levels in the wafer architecture. As mentioned previously, the major attribute of MNOS is its retention of data without standby power. It is, however, not sufficient to control power only at the memory array. The controller circuits,

even though their numbers are not as large as the number of MA's, would consume an inordinate amount of power without power management.

For power considerations, each of the three circuit types, CC, AC, and MA, are divided into two sections: the "address compare" with its associated circuitry and the balance of the unit. In figure 2-5, the address compare circuitry is shown as the rectangle within the respective unit interfacing with the corresponding higher level bus. Of the several CC's fabricated in the quadrant, only two operable ones are connected to the power bus at wafer probe time to reduce power consumption. When power is supplied to the quadrant, only the two CC address compare circuits consume power. If one of these CC's should be activated through the Z-bus by the system controller, that entire CC is powered up, which causes power to be supplied via the Y-bus to all AC address compare circuits in that quadrant.

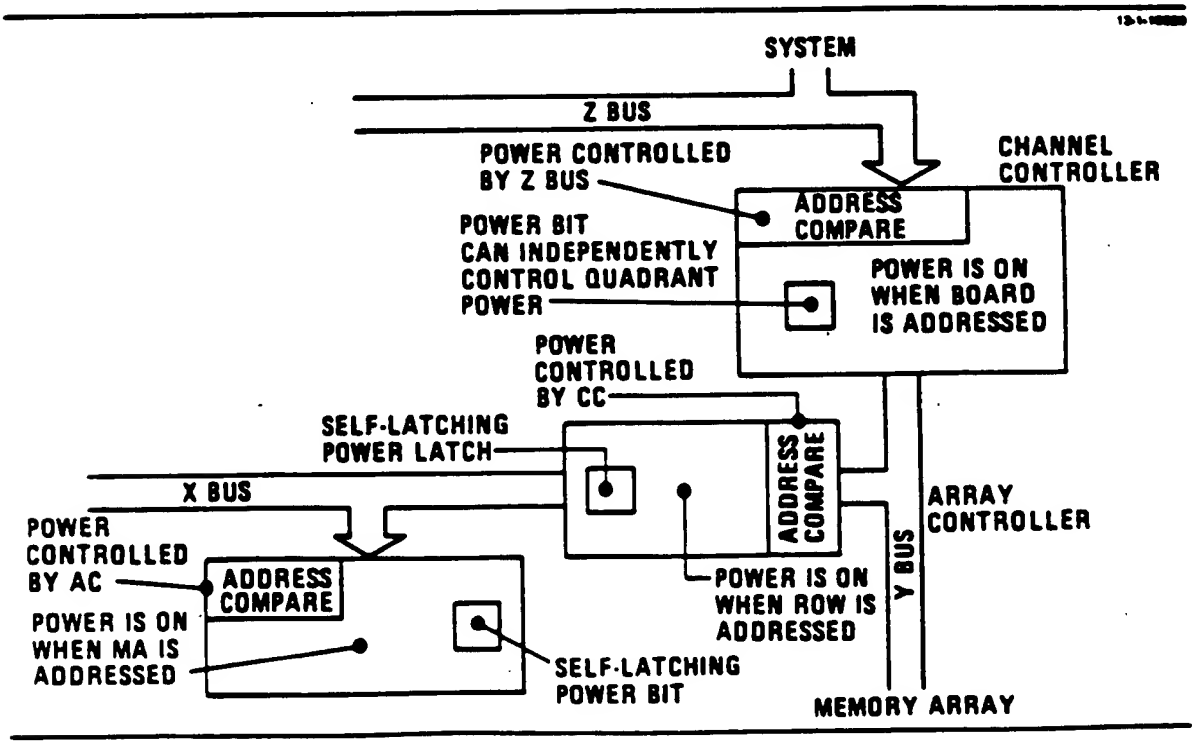


Figure 2-5. Power Switching

LEX15434

When one of the AC's is activated through the Y-bus, the remainder of the AC circuitry is powered up; this causes power to be supplied via the X-bus to the compare circuits of all the MA's in that row. In turn, an addressed MA will have power supplied to the remaining portion of its circuitry, which includes the 4096-bit memory array itself.

The net effect is that for all the MA's on the wafer, only those active arrays and their respective controllers dissipate power. The visual image is then of a narrow path of power flow that is first vertical then horizontal. It remains for a few milliseconds; it is then changed as access is required of another array. This constantly changing, darting flow of power is distributed over the entire wafer and results in a close to uniform distribution of heat on the wafer, dependent only upon the distribution of usable components.

Since, in the case of writing data, an extended write signal is required, it is necessary to supply array power for the entire write time, while data are being written into other arrays in different quadrants. Additionally, this capability must be achieved with a minimum amount of power consumed. To this end, a power bit flag is provided each CC. When set, the power flag in the CC provides power to only those MA's that are set for the extended write mode. The write mode is terminated for all MA's in the quadrant by resetting the power bit flag in the CC.

Power to each of the wafers is controlled by off-wafer power switches. One of these power switches can switch a group of 32 wafers. This switching of power to wafers in groups allows the power consumed by the CC associative decoders to be minimized. These off-wafer power switches, as well as those on-wafer, are controlled by the addresses generated by the system controller.

2.6 SYSTEM OPERATION

Immediately after power is switched to the memory system, the power kernel of the system controller receives the system enable signal and begins memory system initialization operation. The first step the system takes is organizing

the system controller into a working configuration. This is performed in three steps; first, the power kernel organizes itself into a functional group of processors; then the power kernel enables an operational system microprocessor CPU; and finally the system microprocessor CPU finishes configuring the rest of the system controller. System initialization is completed when the system microprocessor has configured the error detection/correction circuitry, and fetched and refreshed the proper address tables.

Once system initialization has been completed, the memory is ready to execute a system command. This system command is either for a data file operation, read, write, or erase, or the periodic built-in-test. The periodic built-in-test scans the memory wafers and removes any faulty components. This test is used, only occasionally, to prevent multiple failures from occurring in the less frequently used areas of the memory, e.g., the spare memory arrays, to minimize the probability of losing data. Once the periodic test is completed, data concerning any faulty components is sent to the external system.

The data file operations are the basic functions of the system; they manipulate the data stored in the MA's. The command for the file operations specifies the type of operation, whether it is erase, write, or read, and the data file which is to be operated. The system controller then fetches the addresses used for that data file. Next, the system controller initializes the high-speed sequencer to generate the proper sequence of memory wafer commands. The high-speed sequencer is then enabled and the first pair of quadrants is addressed. Shortly after the quardants are addressed, the data transfer, if any, begins.

During operations which involve data transfer, the high speed sequencer also issues commands to the error detection/correction circuitry (EDAC). The EDAC encodes the data being written into the memory with a single error-correcting/double error-detecting (SEC/DED) code. When data are being read out of the memory, the EDAC decodes the SEC/DED coding bits and uses them to identify and correct all the single bit errors in a group of sixty-four data bits. Should an error be detected during a read, the EDAC issues an error status message for the system microprocessor which indicates which bit is faulty. This information

is used by the system microprocessor during reconfiguration. When an error status message is received by the system microprocessor, it sets a status flag which indicates that reconfiguration is necessary. This reconfiguration only occurs when the reconfiguration system command is received.

Once a file operation has been completed, the memory is ready for another system command. This command can either be another file operation or the power down operation. However, if the reconfiguration request status bit is set, the reconfiguration command should be issued, especially if another file operation is not to be executed. The reconfiguration command causes the system microprocessor to analyze the error status data to determine the component most likely to have caused the error. The questionable component is then issued a sequence of test commands by the high speed sequencer. The system microprocessor monitors and analyzes the responses of the component for each test step until the operability or non-operability of the questionable component is established. Once the fault has been verified, the system microprocessor locates a spare and replaces the faulty component. After all the detected faults have been removed, the system microprocessor issues status information to the external system which indicates those components that have failed.

The final command to the memory system is always the power down command. This command provides for an orderly power down for the memory system. The system microprocessor is given time to store all of its addressing and status tables, and then power down the memory wafers in the proper sequence. Once the power down command has been completed, power can be removed from the system without any repercussions.

Section 3
HARDWARE DESCRIPTION

3.0 SYSTEM HARDWARE

The memory system architecture consists of the wafer memory modules and the off-wafer control circuitry. The main components of the off-wafer circuitry are the system controller and the error detection/correction circuits. The wafer memory modules consist of the channel controllers (CC's), the array controllers (AC's), the memory arrays (MA's), and the buses which connect them.

The off-wafer system controller supplies the intelligence of the system. It not only interprets the system commands and supplies detailed instruction sequencing to the wafers, but also monitors the operation of all system components. Should a fault be detected, the system controller executes a built-in-test to isolate the fault, and then reconfigures the system to remove the faulty unit.

The off-wafer error detection/correction circuitry performs the encoding/decoding and error correction of the system data. In conjunction with the system buffer memory, the error detection/correction circuitry under programmed control also performs a bit transformation of the encoded data to ensure that each bit of the 72-bit word is stored in a different memory array.

The channel and array controllers implement the memory addressing by selectively switching data, control information, and power to the intended group of memory arrays. This addressing technique minimizes the power consumption of the memory wafers, which constitute the bulk of the memory system.

3.1 MEMORY ARRAY

The basic component of the system is the 4k-bit MNOS transistor memory array. The most significant feature of the memory array is its ability to retain data without any standby power. A memory array is composed of 4096 MNOS transistor memory cells, arranged in a 64 x 64 matrix, and NMOS transistor peripheral and control circuits. The control circuits consist of three main sections: the data circuits, the addressing circuits, and the chip instruction and timing control circuits (depicted in figure 3-1).

The MNOS transistor memory matrix occupies approximately 40 percent of the memory array area, which measures approximately 17.3 mm^2 (26.9 x 10^3 $mils^2$). The NMOS data circuits associated with the memory array occupy almost another 30 percent of the area; the addressing, instruction decode, and timing control circuits make up the remaining 30 percent.

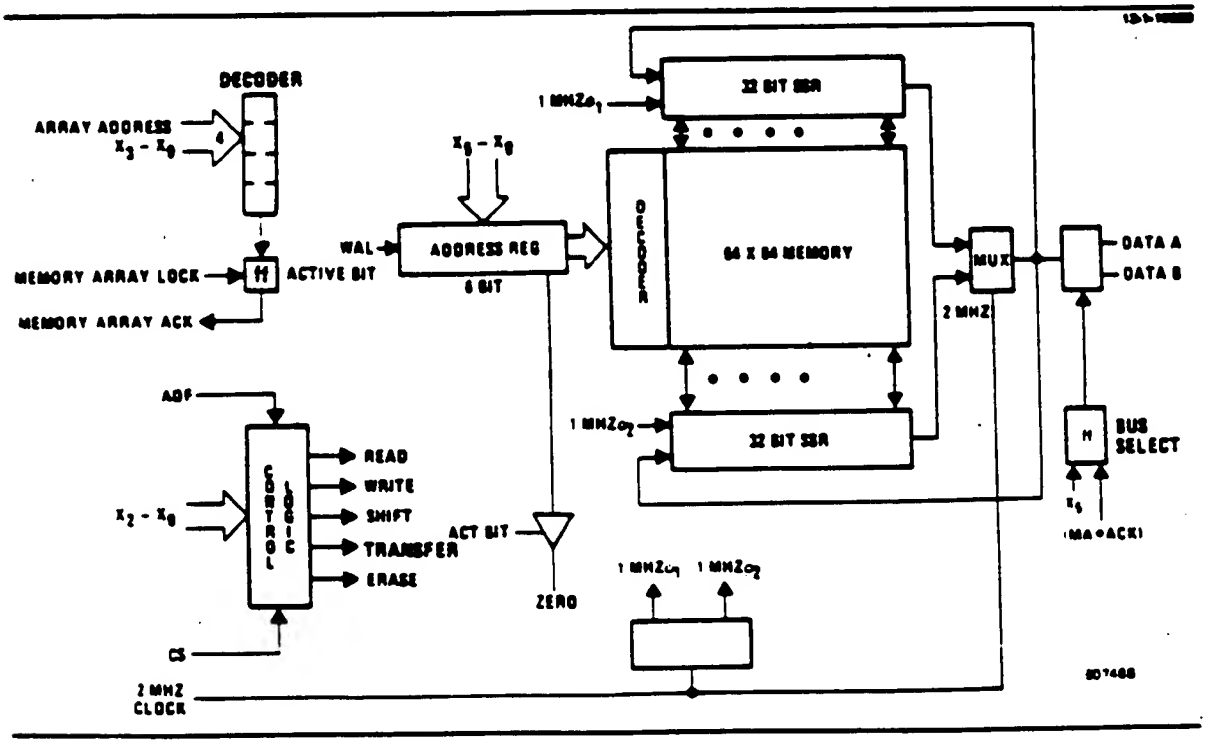


Figure 3-1. Memory Array

CCL unlock (CUL) causes all previously locked CC's to unlock, independent of any CC addresses. A CC which has been unlocked has its active bit cleared, causing power to be shut off in both the CC itself and the associated Y-bus. When this Y-bus loses power, the rest of the quadrant, including all the AC's and MA's, also loses power. However, there is a power bit in the CC which is set by a CC command. When set, this power bit maintains power to the Y-bus, and all lower levels of the quadrant, even after the CC has been unlocked. The CC power bit, in conjunction with the AC self-latching power switch and the unclearable MA active bit, allows a quadrant to achieve a minimal power state where only the CC decoders and the MA's are fully powered.

The purpose of this elaborate power switching scheme is to compensate for the additional power used during the extended write and erase operations. When the write voltage is applied to an MA for 175 microseconds, or the erase voltages for 2 ms, no additional data or control is needed in the quadrant, so the CC's, AC's, AC decoders, and MA decoders cease to perform any useful function. These components can then be forced into a low power state, through a combination of CC and AC unlocks, with the CC power bit set. Once the write or erase operation has been completed, the CC power bit is cleared and the entire quadrant shuts down on the ensuing CC unlock. Since a read operation ends after the data is shifted out of the quadrant, the CC power bit is not set and the quadrant shuts down upon the CC unlock.

The number of CC decoders which are powered at any one time is kept to a minimum by switching power to groups of wafers. This wafer group power switching is performed by the system controller during addressing, and only those wafer groups which contain the current data file are powered.

3.5 WAFER SUBSYSTEM

Figure 3-12 illustrates the system in its physical relationships, particularly with respect to the five memory module canisters. Each canister is separately connected to the system control I/O board. From this connector the wafer stack bus runs the length of the canister with each wafer board soldered to it.

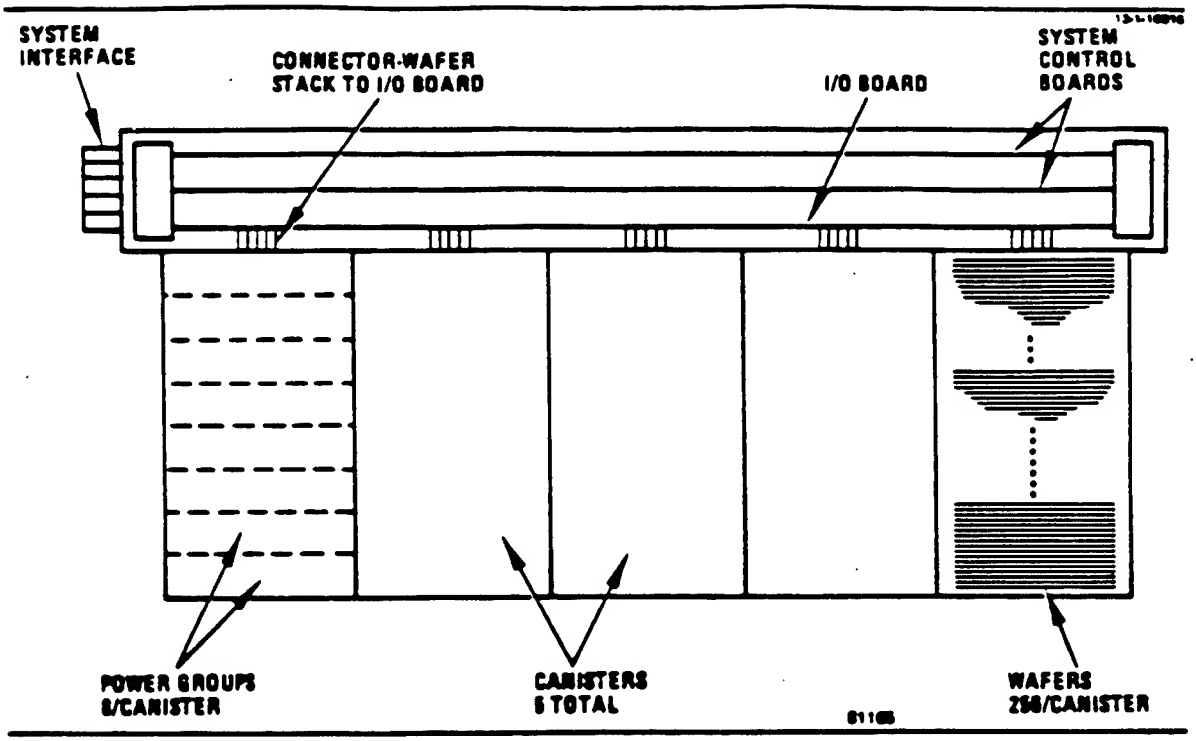


Figure 3-12. System Structure Block Diagram

Therefore, its total length is just that of the canister itself. The system control is located on two boards just above the I/O board and interconnected via connectors on either end. The external system interface emanates from one of these connectors.

The memory is composed of five memory module canisters containing 256 wafers each with an average of 0.5×10^6 bits of nonvolatile memory per wafer. To conserve power and minimize peak power requirements, each canister is subdivided into 8 independently controlled power groups such that only one power group out of the total system is supplied power at any one time. From this restriction, it thus follows that the 18 data blocks which comprise a single file, are located within the 32 wafers of one particular power group out of the total of 40 power groups.

The power control circuits are located outside the canister itself on the PC board associated with the system control I/O. Two power control circuits are required: one for V_{D1} and one for V_{DD}. In each power control circuit are eight separate power lines, only one is in electrical contact to any particular wafer. Under system control, power is selectively applied to one bus at a time. Thus, power is applied to only 1/8 of one wafer stack (32 wafers) at a time.

Operation is illustrated for the case of writing a file wherein six wafers are required to be operational at any one time. Power is applied to the power group where the file of interest lies. This is represented in figure 3-13 with power applied to power group 4 and canister No. 5. Shown are six selected wafers where data is being written. Each wafer draws approximately 2 watts. Each of the remaining 26 wafers of that power group draws only approximately 0.1 watt. (The associative decoders of the 8 CC's are powered.) No power is drawn by the wafers in the remaining power groups of this canister. Approximately every 100

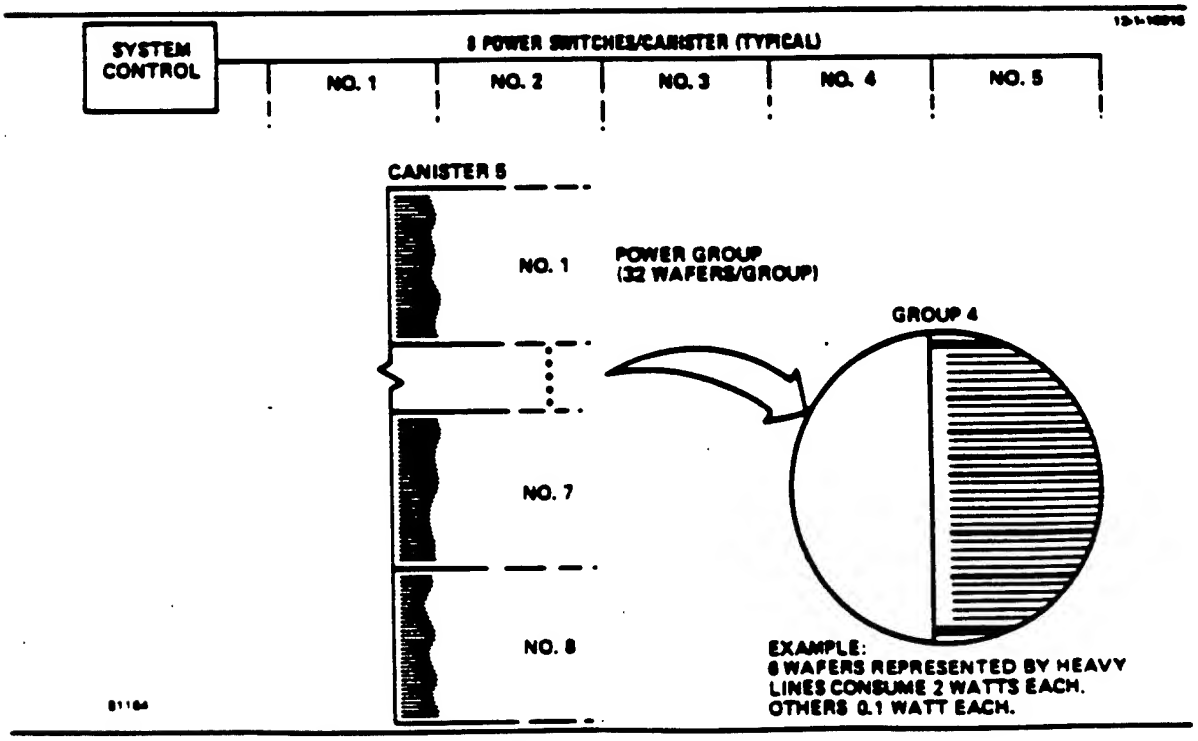


Figure 3-13. Wafer Write Power Requirement

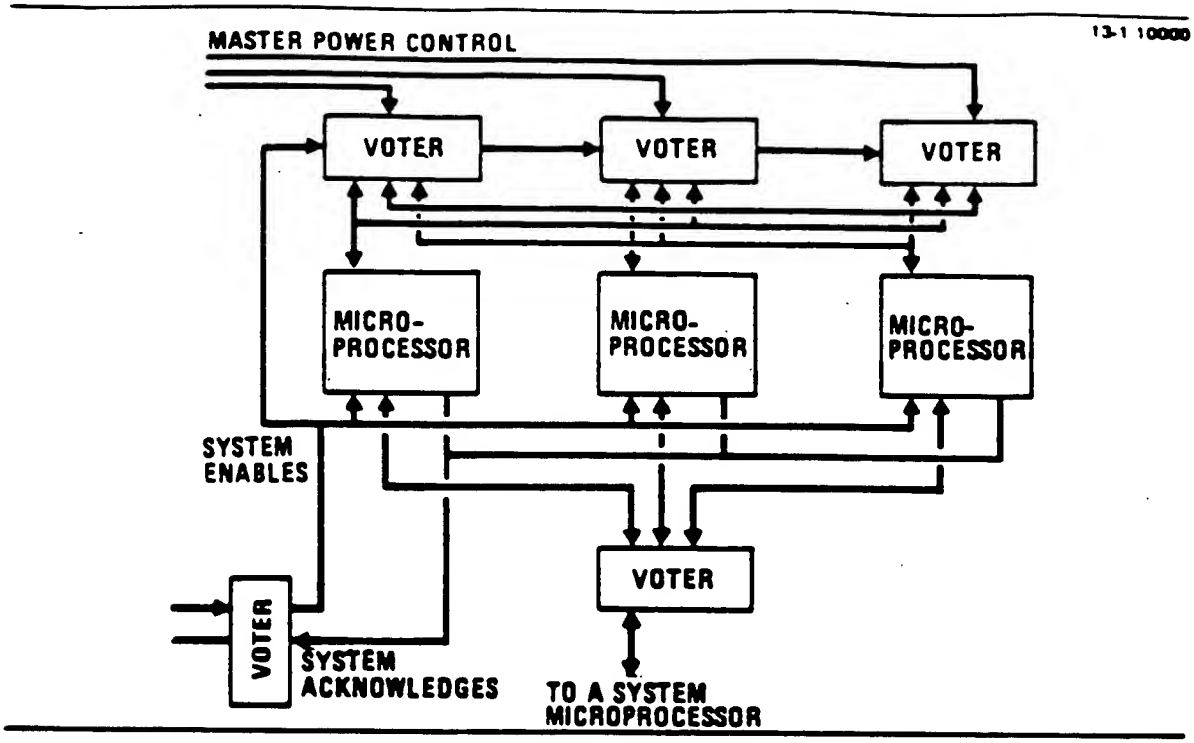


Figure 3-14. Power Kernel Organization

by their NVM status latch, to power-up and initialize the entire power kernel. If a majority of the n-active microprocessors are functioning, an acknowledge signal is sent back to the external system. The external system, being confident that a workable power kernel configuration has been achieved, pulses the GO interrupt and the power kernel, replaces all of its faulty microprocessors, and proceeds to start the system bootstrap.

Should the external system conclude that a functioning power kernel configuration has not been achieved, because no acknowledge was received or for any other reason, the external system continues automatic configuration by clearing the system enable and setting the second system enable. This second system enable switches power to the microprocessors designated as second active. This second level of NMR configuration sends a first acknowledge, and then reconfigures the entire power kernel so that they are now designated as active. This new active NMR group sends a second acknowledge so that the external system can switch back to the regular system enable.

Should the second level of NMR configuration prove inoperative, the external system can still attempt to achieve a workable NMR configuration by brute force. This is accomplished by a set of master power control lines. There is one of these master lines for each power kernel microprocessor. When set, power can be directed to any single processor exclusive of its NVM status, or any other local control. Similarly, power can be removed from any microprocessor by these lines. In this manner, one configuration is achieved. Then each of these processors works to reconfigure the power kernel until each microprocessor has its proper NVM status. The external system can then return configuration control to the automatic mechanisms of the power kernel using the system enable.

The power kernel alters its configuration, under automatic control, similar to the way the external system controls the configuration of the entire power kernel. Once an operative NMR configuration is activated, the good microprocessors will replace any of the microprocessors determined to be faulty. The active processors can address and switch power to any of the inactive processors at the various status levels. For each processor of an NMR configuration, there are other spare processors at each of the four status levels: active, second active, standby, and dead. To replace an active processor, the second active processor is powered and tested. If operative, the second active processor is switched to active status while the previously active faulty processor is switched to dead status. Once a processor has been switched to dead status, it can only be revived by external system control of the master power lines.

The status of the standby processors is then updated so both an active and second active microprocessor is available to each processor component of the NMR configuration. The detailed control of the status switching is accomplished by power enable and disable controls which are addressed by the operative NMR configuration.

The power kernel is immune to virtually all transient faults. The external outputs of each microprocessor of the operative NMR configuration tolerate

transient faults by being piped through a set of hardware majority voters. Transient faults internal to the power kernel, such as test results and testing comparisons, are eliminated through software majority voting, and the power and status control are all performed through hardware voting. Since delegation of a microprocessor to dead status is a drastic step, no microprocessor is assigned a dead status until it has failed at least two test procedures. However, the master power control lines of the external system can always revitalize the microprocessors mistakenly assigned dead status, should the situation ever arise.

The software voting procedure used by the power kernel processors to configure themselves into a fault-free configuration proceeds as follows. Each processor executes a test routine upon receiving the GO interrupt. The result of this test routine is a number which reflects the current operational capability of the processor. This number is a form of checksum. Each processor receives the test result from all the other active processors and compares them. Any processor's test result which differs from the majority of the others is assumed to be faulty. So after a repeat of this software test-and-vote procedure, the faulty processor is replaced.

Communication between the processors is accomplished through their universal voters, shown in figure 3-15. The status number of each processor is sent to the others by each processor loading a register in the respective universal voters. The registers in each voter allow this transfer to be made simultaneously by all processors to eliminate differences in each processor's program. Then each processor reads out the status numbers out of its voter. Each number is checked against the others. After each comparison, the processors output a power switch control word to the universal voter of the processor just checked. Should this processor's status number not agree with at least one of the other two, the checking processor outputs a power-off control word. Otherwise, the processor outputs a power-on control word. The voting network of the universal voter then sends the control word of the majority to its power switch, which then either shuts off or maintains power to its processor. Each processor, in turn, is power switched according to the majority decision.

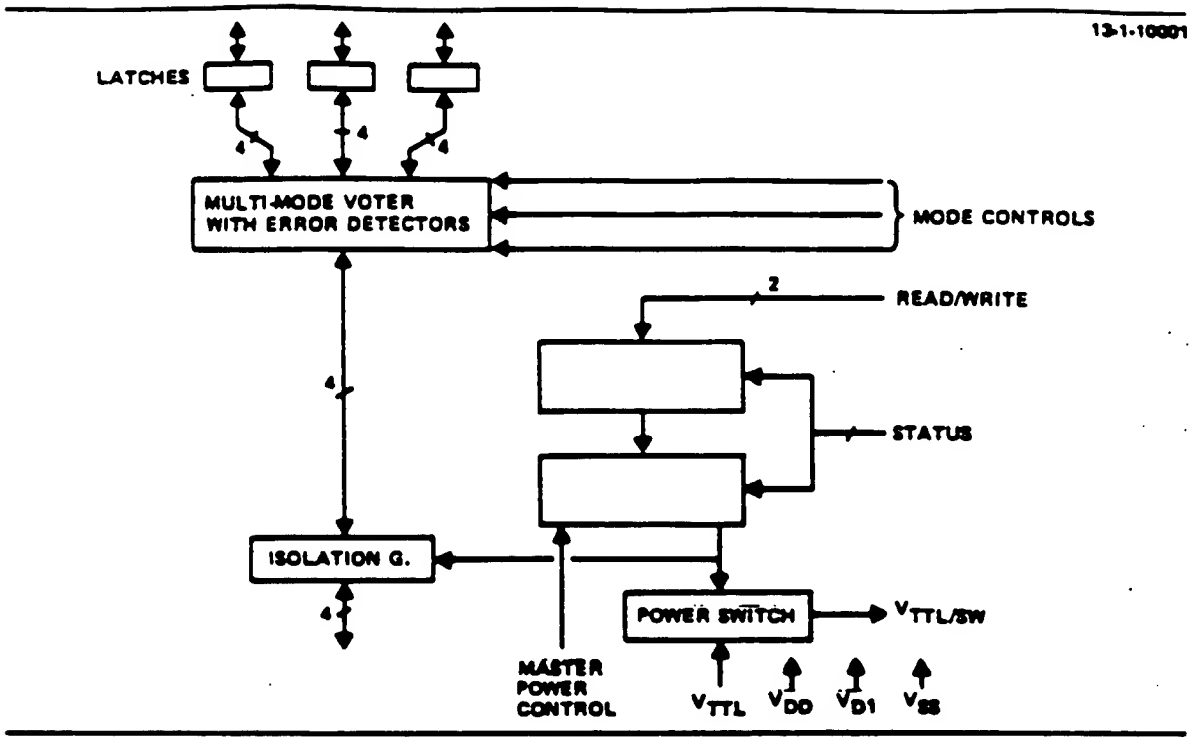


Figure 3-15. Universal Voter Configuration

Should the majority determine that a processor is faulty and so power it down, the remaining processors power-up a standby spare via this spare's universal voter. The spare processor next to become active is chosen according to the status word stored in the voter's NVM. However, should a reliability analysis show that this dependence upon the universal voter's power switch and status word is too great, an alternate power switching technique is being examined. In this alternate technique, a faulty processor is only powered down; the replacement of the failed processor is left to the external system. The processors would send out a signal to the external system which would cause a new processor to be activated as a replacement via its master control line.

The NVM status register that controls the power switching is contained in the universal voter chip. Power switching is accomplished by requesting the second

standby processor to be powered instead of the active processor. Then the processor whose NVM status is set to second standby is powered. Finally the status of all the other processor's have their status updated in such a way that one is active, one is second active, and at least one is spare (if enough remain). The faulty processor is then relegated to a dead status. Thus, each processor of the power kernel can be individually addressed by its NMR location and status. A similar technique is used for power switching throughout the off-wafer subsystems.

Upon completing its self-test and reconfiguration routines, the now fully operable power kernel begins a test of the system microprocessor CPU. This is also accomplished through the universal voters, associated with a system microprocessor CPU.

The majority of the power kernel processors vote to issue an interrupt to the CPU under question, to initiate the CPU self-test. The result of this test is a CPU status number which is received by the power kernel processors, via the CPU's universal voter. Each processor then compares this CPU status number with the correct known value. Should the status numbers match, the processor will send a pass signal to the CPU's voter. When the majority of the power kernel processors pass the system microprocessor, it receives another interrupt and begins a bootstrap of the rest of the system controller. If the majority of the processors fail the CPU, it is powered down and a replacement powered up, as was done for a faulty power kernel processor.

The reliability requirements of the power kernel demand that it be composed of only a few chips with short and simple software routines. To accomplish this, the universal voter should be developed as a custom LSI chip. The processor will also be a single chip microcomputer with on-chip ROM and RAM. Using a small number of LSI chips not only increases the subsystem reliability, but also reduces its power consumption. To reduce the power even further, CMOS technology will be used wherever possible, and CMOS/SOS technology will be used, if possible, to improve the power kernel speed. RCA is developing an 8-bit single-chip microcomputer, 1804, which uses CMOS/SOS technology. This

combination of the high speed, low power CMOS/SOS technology, with on-chip ROM and RAM, makes the 1804 especially desirable.

The software of the power kernel is basically straightforward and simple. By properly wiring the components of the power kernel, all differences between the routines in each processor can be eliminated. The key factor is making all status number comparisons the same for each processor, because a processor does not know whether it is voting on its own or another processor's status number. Using the same routines in each processor simplifies the testing and verification of the power kernel software.

3.6.2 System Microprocessor

The system microprocessor is a sophisticated high-speed processor which performs the main control functions of the memory system. The system microprocessor functions are:

- Responding to memory system commands,
- Updating the memory system status word,
- Issuing commands to the high-speed sequencers,
- Monitoring the high-speed sequencer function status,
- Monitoring the error detection/correction status,
- Initiating and monitoring all built-in tests,
- Performing system fault diagnosis, and
- Controlling system reconfiguration.

The system microprocessor configuration is shown in the block diagram of figure 3-16.

The system microprocessor is basically an interrupt-driven device. Normally the processor executes a monitor program which watches the system interface and the error detection/correction circuitry interface, as well as issuing commands to the high-speed sequencer. The other functions of the subsystem are executed in response to interrupts from the other memory subsystems. An interrupt from

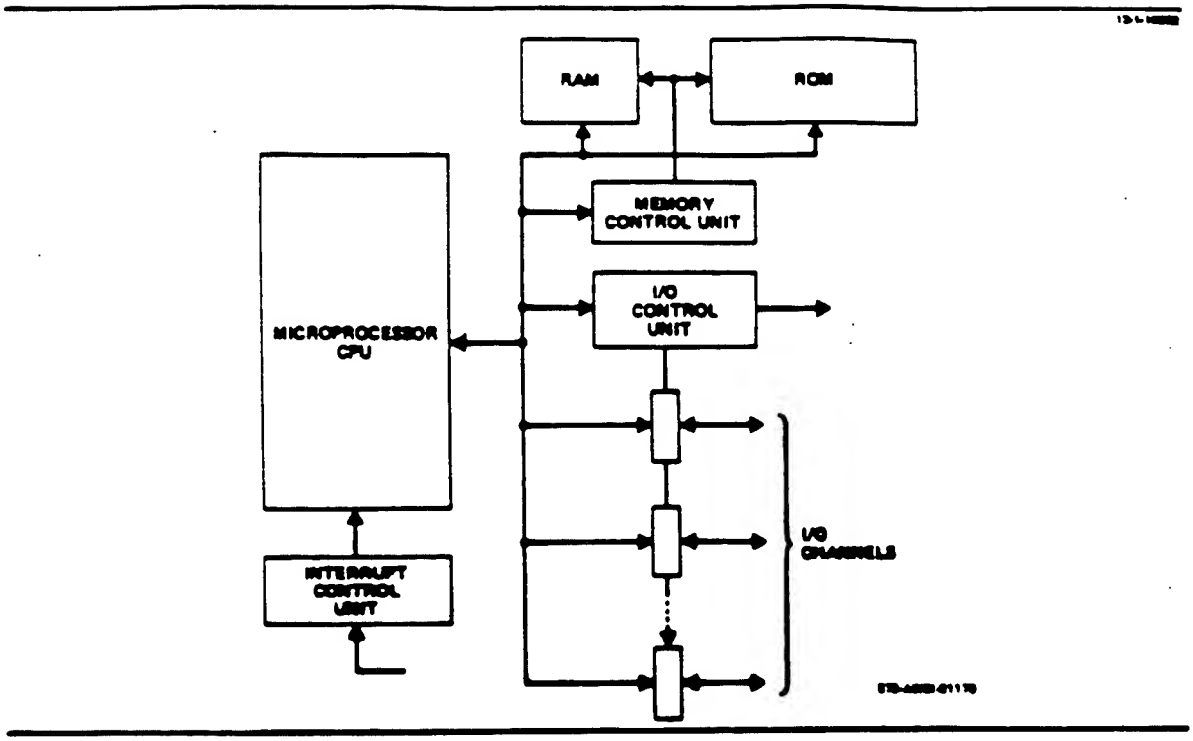


Figure 3-16. System Microprocessor

the power kernel causes the system microprocesor CPU to perform a detailed self-diagnosing test which informs the power kernel if the microprocessor CPU is properly functioning. Interrupts from the high-speed sequencer cause the microprocessor to perform tests of the wafer controllers since an error has occurred during memory wafer addressing. Interrupts from the error detection/correction subsystem (EDAC) indicate a data error and cause diagnostic analysis of the status word to be performed to determine which built-in test is necessary. The results of most built-in testing is interrupt-based and causes some system reconfiguration. The software of the system microprocessor is given in section 4.

The diagnostic analysis and built-in testing done by the system microprocessor require variable amounts of RAM as scratch pad memory. To reduce power requirements of the processor subsystem, the 4kb of scratch pad is powered up in partitions of 512 words, as needed. Since an interrupt-driven system executes

mostly modular routines, the 28kb of program ROM can be similarly partitioned into 2k sections which become powered when the corresponding routine is executed. The memory control unit performs this power switching of the memory partitions as well as supplying the memory control signals.

The interrupt-based nature of the system microprocessor requires a multilevel prioritized interrupt structure. The interrupt controller supplies the necessary decoding and prioritization of the interrupt signals from other subsystems as well as information specifying the nature and origin of the interrupt requested. The priority of an interrupt depends upon how critical the situation requiring the interrupt is. Error and malfunction interrupts have the highest priority, because they lead to other situations which may result in other interrupts. For example, since an addressing error interrupt eventually results in data error interrupts, addressing errors are handled prior to data errors. Such an interrupt structure requires some intelligence in the other subsystems since they must continue to carry out their basic functions, if possible, while they wait for their interrupt to be acknowledged.

Several commercial 16-bit microprocessors have the speed power products necessary for the system microprocessor, as well as the required sophistication in instruction set and I/O interrupt capabilities. Three currently existing 16-bit microprocessors which fit these requirements are: the Fairchild 9440, the Intel 8086, and the Texas Instruments 9900A. Because these processors have only recently been developed, no final selection can be made until the development of all three is completed. A comparison of each of these is made in table 3-7.

Because communication between the system microprocessor and the other components of the system controller, the error detection and correction (EDAC), and the memory wafers is critical, reliability may dictate that a redundant busing system be used throughout the off-wafer subsystems. Although the level of redundancy will depend on the results of a detailed reliability analysis, the envisioned implementation of the redundancy is to switch each bus as a unit. The bus unit will consist of both the bus lines and the bus transceivers at each end. This redundant configuration simplifies its implementation and reduces bus switch complexity.

TABLE 3-7. MICROPROCESSOR COMPARISONS

MICROPROCESSOR	TI 9900A	INTEL 8086	FAIRCHILD 9440
MAXIMUM CLOCK	3 MHZ	5 MHZ	12 MHZ
REG TO REG ADD TIME	> 12 μs	1.5 μs	1.25 μs
POWER – CPU	0.5W	0.8W	1.0W
POWER – SYSTEM	< 4.9W	< 5.2W	< 5.3W
REGISTERS	ALL IN MEMORY	8 ON-CHIP	8 ON-CHIP
I/O CONTROLS	MSI CUSTOM DESIGN	MOSTLY ON-CHIP	9442 I/O CHIP
INTERRUPTS	> 16 VECTORED	> 16 VECTORED	> 16 VECTORED
INTERRUPT CONTROL	MSI CUSTOM DESIGN	MOSTLY ON-CHIP PLUS 8259A	INTERRUPT CONTROL CHIP – MAY NOT BE DEVELOPED
ADDITIONAL CHIPS	NONE	8224 CLOCK CHIP	MAY REQUIRE 9441 MEMORY CONTROLLER
TOTAL CHIPS	≈ 30	≈ 20	≈ 20
INSTRUCTION SET	GOOD BUT SLOW	8080 CAPABILITY AND MORE INCLUDING MULTIPLY AND DIVIDE	NOVA INSTRUCTION SET
TECHNOLOGY	I^2L	HMOS	I^3L
TTL COMPATIBLE	REQUIRES SPECIAL INTERFACE	YES	YES
DEVELOPMENT STATUS	IN PRODUCTION	SAMPLES AVAILABLE	STILL IN DEVELOPMENT

Reliability also dictates that some amount of redundancy be used in the system microprocessor implementation. The system microprocessor will be sectioned into a set of modules which all have unpowered redundant spares. The modules will consist of the microprocessor CPU, the memory and memory control unit, and the I/O and interrupt controllers. All of the modules, except the microprocessor CPU, will be tested and reconfigured under control of the microprocessor CPU. Testing and control of the microprocessor CPU will be governed by the power kernel.

Once the power kernel has initialized a properly functioning microprocessor CPU, the CPU bootstraps the rest of the system microprocessor so a fault-free configuration is activated. Then the system microprocessor initiates and tests the high-speed sequencer (HSS) and the error detection/correction circuitry to complete the activation of an entirely fault-free off-wafer control subsystem. The system microprocessor then accepts a system command. The HSS and EDAC are set up to execute this command, and the memory system begins execution.

while the memory system is executing a system command, the system microprocessor is monitoring the progress of the HSS and the EDAC. Should the EDAC detect any errors, the system microprocessor will store the corresponding information about the errors for use during built-in testing and reconfiguration. After the system commands have been completed, the system microprocessor initiates any necessary built-in tests and use the results of these tests to reconfigure the system. The system microprocessor also continuously updates the system status word to indicate both the execution and error states of the entire memory system.

3.6.3 High-Speed Sequencer

The high-speed sequencer (HSS) is a subsystem of limited capabilities whose major function is to issue the specific instruction sequences and timing signals to the wafers. Each instruction stream is initiated by a command from the system microprocessor. The instruction streams which the HSS generates include not only the erase/write/read sequences, but also many of the sequences for testing and reconfiguration of the wafers. Since each of these instruction streams has many similarities, they are generated in blocks of subsequences to reduce duplication. The subsequences generated to make up the entire instruction stream are selected by flags set in the system microprocessor-HSS status word by the system microprocessor, similar to conditional subroutines. Since the HSS must issue different sets of instruction sequences as these flags are tested, the HSS subsystem is organized as a microcoded machine. The microcode controls the data paths and components of the HSS as well as containing the instructions issued to the wafer. Besides issuing the instructions to the wafer, the HSS must also monitor the wafer's responses to the instruction stream, and issue control signals to the EDAC circuitry to maintain some synchronization between the EDAC and wafer during data transfers.

A simplified block diagram, figure 3-17, shows the main sections of the HSS. The microcode given in figure 3-18 is based on the requirements of each of these sections. The central section contains the microsequencer and the microstorage. The microsequencer generates the addresses of the microcode. An

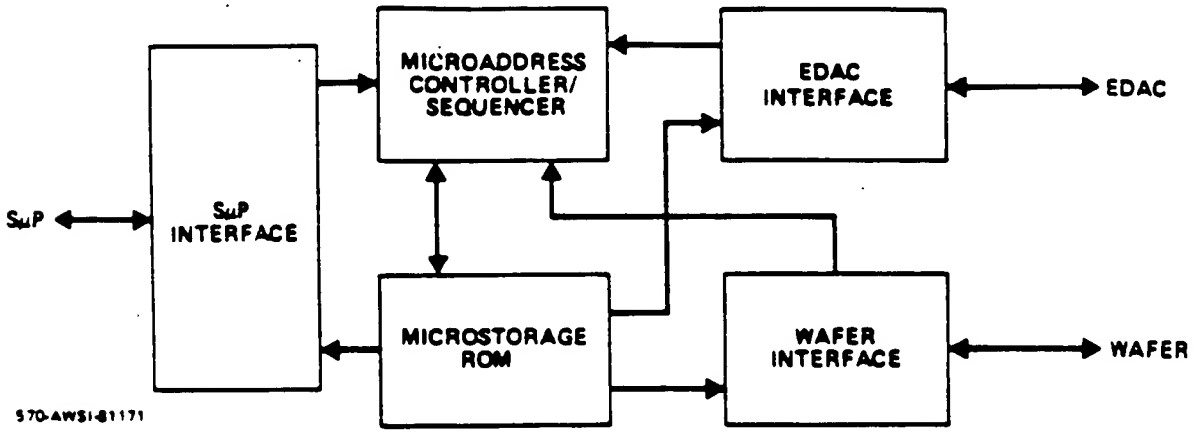


Figure 3-17. Simplified HSS Block Diagram

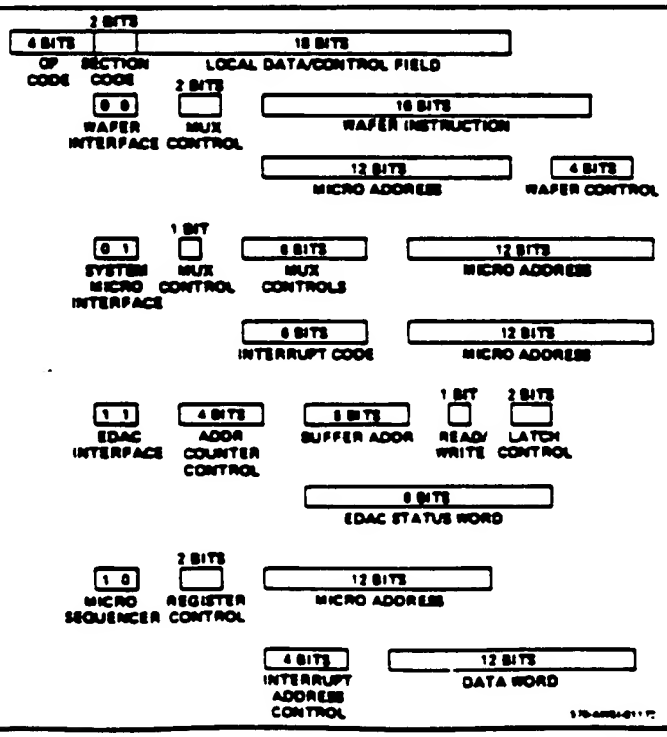


Figure 3-18. Initial HSS Microcode Format

Advanced Micro Devices 2910 microsequencer is used here since it is a single-chip 12-bit microsequencer which fulfills the speed requirements of the HSS. Since so much is necessary to control the instruction sequence sent to the wafer, the microsequencer operates at 8 MHz, allowing 4 microinstructions (3 control microinstructions plus one microinstruction for the wafer) to be executed in the 500-ns cycle time of wafer instructions.

To reduce the power consumption of the microinstruction storage ROM, CMOS ROM's are used. Because CMOS ROM's do not have the cycle speed required, a pipelined configuration is used, as shown in figure 3-19. Four microinstructions are read into the pipeline registers simultaneously. While the next 4 microinstructions are being read, the four pipeline registers are accessed by the microsequencer.

The system microprocessor supplies the starting address of the basic microcode routine for the desired wafer instruction stream. The system microprocessor then selects which of the microcode subroutines is to be included in the basic

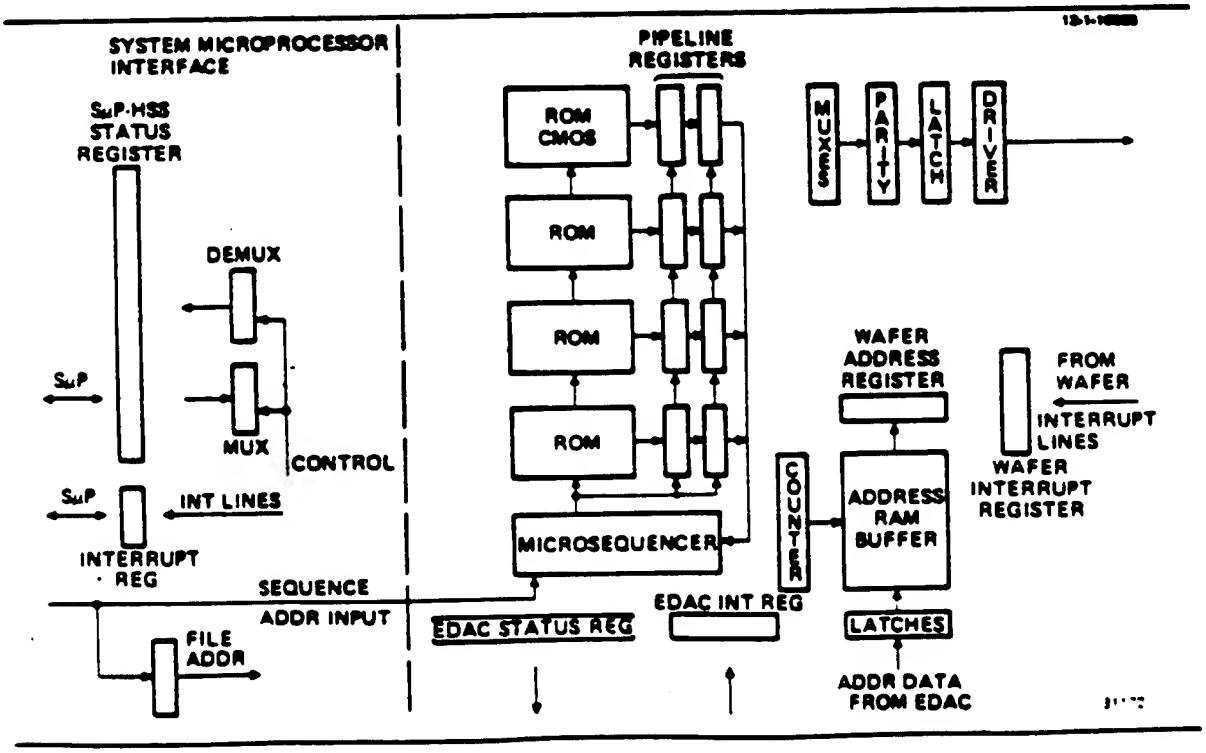


Figure 3-19. HSS Diagram

routine by setting the proper flags of the HSS status register, as will be discussed in section 4.8. These flags signify such things as read, write, or erase, and the number of memory arrays involved. The HSS in turn updates the corresponding flags in the system microprocessor status register so that the system microprocessor can monitor the progress of the HSS and know the current HSS state should an error occur.

The address of the memory arrays for a given data file are stored in a specific memory array on the wafer. The system microprocessor supplies the location of the memory array holding these addresses in the file address register. The HSS obtains the addresses and stores them in the address RAM buffer. Since each of these address words stored on the wafer contains error code bits to maintain their integrity, the EDAC has its data paths arranged in such a way that the code bits can be checked and the correct addresses supplied to the HSS. The HSS sets up the data paths of the EDAC via the EDAC status register. Should any fault situations occur in the EDAC, the HSS is notified by flags set in the EDAC interrupt register. The HSS then notifies the system microprocessor of the situation through the system microprocessor interrupt register.

Since power is only applied to a group of wafers as needed, the addressing information loaded into the HSS address RAM buffer is also sent to the system microprocessor for wafer power group switching. The system microprocessor converts each memory quadrant address into a power switch address via a lookup in the system microprocessor's processor power group table. The system microprocessor then switchs power to the appropriate power group. A data file will be constrained to exist in only one power group to minimize the power used.

Many of the wafer instructions contain addressing information obtained from the address RAM buffer. The content of wafer instructions is dictated by the arrangement of the multiplexer and latch combination of the wafer interface. The arrangement of the wafer interface is dictated by the proper microcode sequence. This microcode sequence not only controls the content of the wafer instruction, but also the destination of this instruction, namely, channel 1 or channel 2. The destination is controlled by setting the CHC (channel control) flip-flops. The CHC flip-flops are encoded to indicate the four channel

access modes, i.e., channel 1, channel 2, both channels, or no channel (all channels closed).

The HSS is the only component of the system which can directly exert such exacting control over the memory wafers. Therefore, its continued operation must be guaranteed. Replication is again used to obtain the required reliability. Here the reliability scheme is to subdivide the HSS into modules similar to the main sections shown in figure 3-11. Each of these modules will have a number of unpowered spares. When the system microprocessor causes the HSS to execute a self-test, it monitors the test results of this HSS execution. Should the system microprocessor then determine a module is faulty, it will switch in a replacement. The exact number of spares will be determined by reliability analysis.

3.6.4 Alternate System Controller Configurations

The main driving forces behind the design of the system controller are high speed, low power, and high reliability. Each of these tradeoffs has been balanced in the current system controller design, which consists of the high speed sequencer, the system microprocessor, and the power kernel. Alternative system configurations, which perform the same functions, will be described, with the impact of each on the previously explained design goals.

The first alternative system controller architecture places more emphasis on the high speed sequencer, by incorporating the functions of the system microprocessor in the high speed sequencer. To enable this modified high-speed sequencer to complete the tasks of both the high speed sequencer and the system microprocessor, it must operate at a very high speed, which dictates the use of bipolar technology. Since the computational power of the system microprocessor must also be incorporated, a 2900-type bit-sliced microprocessor is indicated.

This 2900-based high speed sequencer must have the same capabilities of an 8086-type microprocessor: 65kb address space minimum, 16-bit data words, and

16 prioritized vector interrupts. A 2900 system incorporating these features, including a carry look ahead generator to guarantee operation above 10 MHz (3 to 4 times the speed of the present high speed sequencer), takes 13 major DIP's which consume approximately 11.2W. These numbers do not include memory or I/O circuitry, they only replace the system microprocessor CPU and the high speed sequencer 2910 address controller. The circuitry replaced by this 2900 system only took 6 DIP's and 4.1W; the 2900 system occupies more room and burns much more power. Both the system microprocessor and the high speed sequencer were originally allocated only 8 W total.

Not only does the increased number of DIP's in the 2900 system increase the size of the active high speed sequencer/system microprocessor, but also it decreases the reliability of the system controller. Therefore, the number of spares must be increased to maintain the reliability. Additionally, because the complexity of the increased size may result in a less testable system, the diagnostic checkout of the 2900 based system may take longer even though it is operating at a much higher speed.

The fact that a 2900 is microprogrammed cannot be ignored. Each microprogrammed instruction takes a number of microinstructions; the size of the program ROM may need to be increased. Every such increase in memory size affects the number of 2909/2911 microsequencers in the 2900 system, which again impacts the power, size, and reliability.

This 2900-based system controller can be further modified to incorporate the functions of the power kernel. This can be accomplished by adding some MSI/SSIlogic to divide functionally each 4-bit slice of the 2900 system into a power kernel processor. This switch to the power kernel is controlled by the external
system. Once operating as the power kernel, each 2900 slice would then perform a functional test and compare its results with those of the other slices and vote to remove any faulty slices. Once each slice was deemed functional, the external system would then be signaled to switch the 2900 slices back to a 16-bit processor.

This doubly modified system would reduce the impact of the larger size of the 2900 system; the total number of microprocessors would be reduced. This version of the system controller also concentrates the design problems into one rather complex microprocessor subsystem.

Power would still remain a problem since most of the 0.5 watts allowed for the power kernel would be used by the additional MSI/SSI logic needed to control the individual 2900 slices. Also the size would probably remain increased because this 2900 system must now be the most reliable section of the entire memory system.

Another alternative system controller configuration is obtained by placing most of the high speed sequencer control functions in the system microprocessor. Such a system controller configuration would reduce the high speed sequencer to not much more than a high speed counter and memory. Perhaps a watt or more could be saved by such a simple high speed sequencer.

The additional burden of directly controlling such a high speed sequencer, however, may place too much of a processing load on the current system microprocessor. In addition to monitoring the high speed sequencer's progress, the system microprocessor must monitor the memory wafer responses. Since the wafer responses to addressing occur in short bursts, the system microprocessor's time is fully occupied during addressing sequences. This occupation forces the system microprocessor to accept data error status messages from the EDAC subsystem only via direct memory access. Then, while the memory wafers are shifting data, the system microprocessor can sort and store any data error status messages in its data error file.

During any single data block sequence, the system microprocessor's time is now completely allocated to necessary functions, so subsequence selection, branching control in the high speed sequences is no longer possible. This loss of capability eliminates the modular programming of the high speed sequencer, which results in individual blocks of instruction for many sequence variations.

Although the additional internal control steps of high speed sequencer programs are no longer necessary, the program storage of the high speed sequencer is greatly increased. Therefore, no real component savings is achieved by this modified system controller configuration.

The final system controller variation to be discussed is the combination of the system microprocessor and the power kernel. Here, a number of active system microprocessor CPU's are functioning together. Each of these CPU's performs evaluation tests on one another similar to the procedure of the power kernel. The two main differences of such a configuration are that the outputs of all active CPU's are voted upon and the power consumed by this multiple system microprocessor system is increased proportionally to the number of active CPU's.

The reliability of this system multi-microprocessor is greatly enhanced by this configuration. Virtually all transient faults in the system microprocessor are removed by the majority voted outputs. Also, a total failure of anything less than the majority of the active CPU's will not crash the memory system or interrupt its operation during a memory system command execution. However, this configuration forces the system microprocessor to have the greatest overall reliability, and so results in an increase of not only the number of active CPU's, but also the number of inactive system microprocessor spare subsystems.

The power increase due to the increase of the number of active system microprocessor CPU's is nontrivial. For a TMR-based system, the power increase is 8 watts, for a total system power of approximately 47 watts for a write operation. An NMR system increases similarly as N increases.

Although each of the system controller variations discussed here have some advantages, they all have significant impacts on the reliability, power, and size of the system. Additional variations of the system controller configuration are possible; any permutation of power kernel, system microprocessor, and high-speed sequencer can be made to fit the speed/reliability requirements, and to perform most of the existing system controller function.. However, an additional cost of power and size results for each of them.

Many of these alternative configurations pushes one or another of the system controller subsystems toward its limits. This tends to limit the flexibility of the system for future expansions of speed and memory capacity.

3.7 ERROR DETECTION/CORRECTION AND SYSTEM DATA BUFFER MEMORY CIRCUITRY

The error detection/correction and system data buffer memory circuitry perform the encoding/decoding and formatting of data stored in the memory system. The error detection/correction circuitry converts the serial data to/from an error correction code word which helps maintain data integrity. (A complete description of this technique is presented in Volume VI, Error Control for a Mass Memory System.) The system data buffer memory circuitry converts the encoded data to/from the format used by the memory wafers. Control for the error detection/correction and system buffer memory subsystem is provided mostly by the high-speed sequencer.

This subsystem will be implemented by a few custom LSI CMOS/SOS chips. The CMOS/SOS technology has the needed speed capability and the desired low power consumption. Existing or proposed LSI chips, such as Advanced Micro Devices 2909 error coding chip, could be used instead of the custom CMOS/SOS implementation, but using the custom implementation saves approximately 5 to 6 watts.

3.7.1 The Code

The simplest single error correcting/double error detecting (SEC/DED) code, a linear cyclic block code is used. An SEC/DED code was chosen because the reconfigurable nature of the memory implies that the appearance of multiple bit errors would be very unlikely, and double error correction would thus lead to an unnecessarily complex implementation. However, a code which can also detect some of these multiple errors can readily be implemented without greatly increasing complexity. An SEC/DED code requires a minimal Hamming distance of 4. An SEC/DED code with 72-bit words containing 64 data bits and 8 parity check bits has been selected. This is actually a shortened pseudo cyclic code based on 128-bit words where the extra bits are assumed to be zero. All of the desirable characteristics of the cyclic code are retained by this shortening.

A coded word is generated by dividing that data word, in the form of a polynomial, by a generator polynomial which defines the code. The generator polynomial used for the code is $g(x) = 1 + x + x^6 + x^8$. For serial data, the data word can be encoded by using a linear feedback shift register as shown in figure 3-20. The eight additional bits generated in this shift register are then appended onto the 64 data bits to form the 72-bit code word.

When a code word is retrieved, the decoding process detects all one and two-bit errors, and specifically locates any single-bit error so it can be corrected. The decoding process is similar to the encoding procedure. Rather than generating the parity bits, it generates a syndrome which identifies the detectable errors. If no errors occur, the syndrome is identically zero. If a single error occurs, the syndrome indicates which bit is in error. After correcting this bit, the resulting syndrome becomes identically zero, indicating no further errors. If the syndrome does not become zero after correction, or never indicates a specific bit was in error, a detectable multiple bit error has occurred. These multiple bit errors cannot be corrected by the code alone, but may be resolved during reconfiguration.

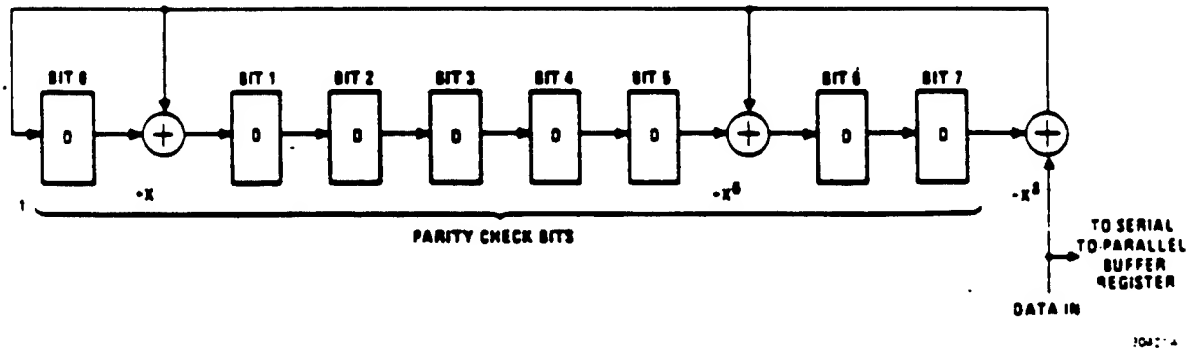


Figure 3-20. Linear Feedback Shift Register Implementation for the Pseudo-Cyclic Block Code Derived From $g(x) = 1 + x + x^6 + x^8$

3.7.2 Encoding Implementation

As data are shifted into the memory system, the data path multiplexer directs the data both into the 64-bit information register and to the linear feedback shift register (LFSR), as shown in figure 3-21. During the 64 shifts which fill the information register, the LFSR cycles each new data bit into the shift register section via input taps. Each input tap corresponds to a term in the generator polynominal, $1 + x + x^6 + x^8$, see figure 3-20. At each tap, the new bit is exclusive-OR'ed with the preceding term to form the input to the next tap. After all 64 bits have been shifted in, the properly encoded 8 parity bits are in the LFSR and all 72 bits are sent in parallel to the system buffer memory.

In order to allow the EDAC to better detect certain types of errors, each 72-bit word is designated as odd or even. Every even word has a 1's cover sequence modulo 2 added to it as the data is shifted in. This cover sequence simplifies the detection of hard stuck-at-one (stuck-at-zero) faults in a group of zeros (ones).

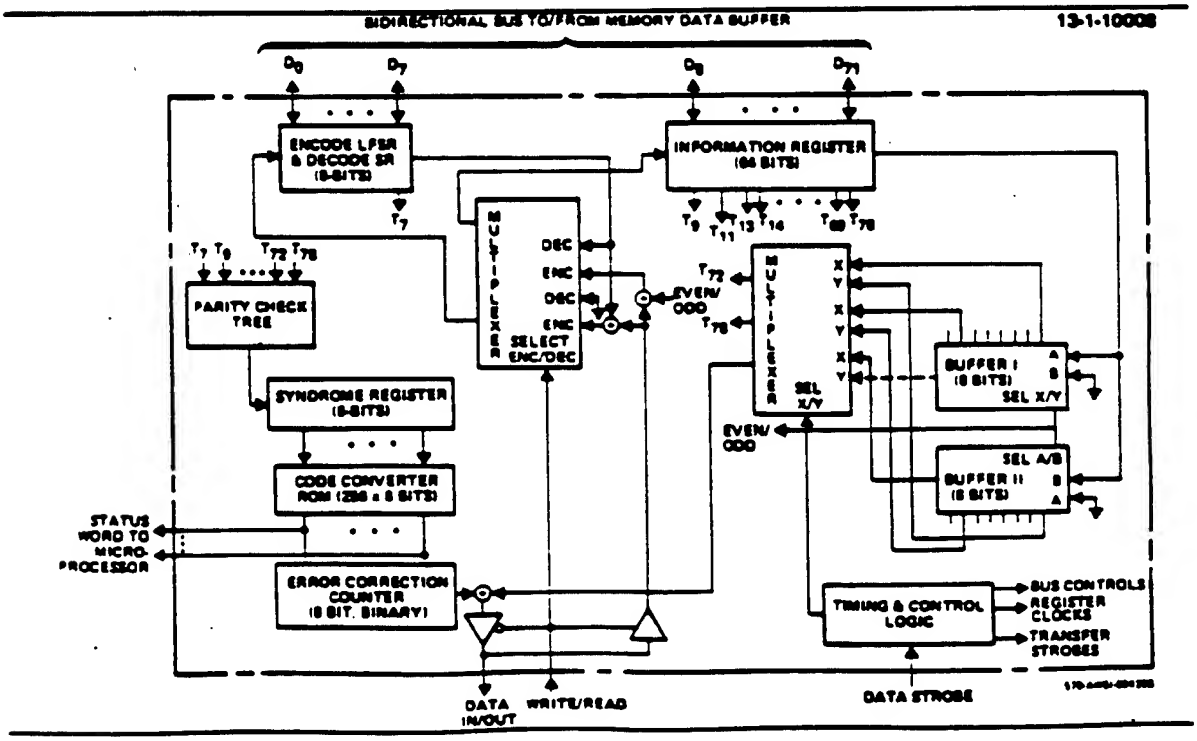


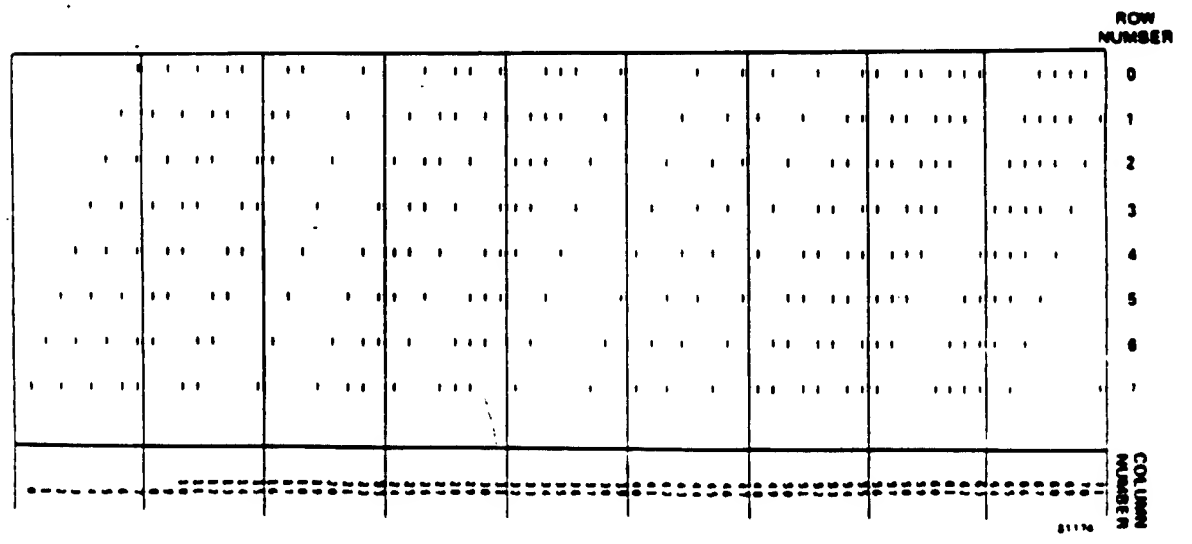
Figure 3-21. Block Diagram; Pseudo-Cyclic Parallel Parity Decoding At 5 MHz I/O Data Rate

3.7.3 Decoding Implementation

Since stored data are received by the EDAC in parallel form, a simplified implementation can be achieved by employing parallel parity decoding. In parallel parity decoding each parity bit is generated by simultaneously summing the data bits which form that parity term. In the shortened cyclic code used, the same parity tree can generate each parity bit. Table 3-8 indicates the bits which are summed for each parity term and shows that each succeeding parity bit can be generated by right shifting the data word one bit, with zeros filling in at the left. After only 8 shifts, the entire syndrome has been generated.

As the data are shifted out of the LFSR and the information register, they are sent to one of the two 8-bit registers, shown in figure 3-21 as buffer I and buffer II. This pair of buffers allows continuous data transmission and decoding after the initial 8 shifts of the first word. After an odd (even) word

TABLE 3-8. PARITY CHECK MATRIX



has had all but its last 8 data bits shifted out through buffer I (II), the following even (odd) word is loaded into the LFSR and information register. During the next 8 shifts, the remaining data bits of the odd (even) word are transmitted while the even (odd) word has its syndrome calculated and buffer II (I) loaded. As data leaves buffer II, the all-ones cover sequence is removed.

3.7.4 Error Correction Implementation

The syndrome calculated by this particular parallel parity decoding method cannot be directly used to perform error correction on the data in the usual fashion. The syndrome is first converted to a useful form by a specially encoded ROM look-up table. The calculated syndrome provides the address to the ROM; the resulting output corresponds to the column number, of the syndrome, as indicated in table 3-8. The resulting output of the ROM is loaded into a countdown counter. Should this counter reach zero during the 64 output shifts of the current data word, the particular bit being shifted out would be complemented; the syndrome indicating it was in error.

The same word loaded into the counter is also sent to the microprocessor as an 8-bit status word. The significance of each status word is shown in table 3-9. When a status word indicating an error occurs, an interrupt for the system microprocessor is generated by the EDAC which sets its error flag. The system microprocessor acknowledges the interrupt by resetting the flag.

Whenever a data error occurs, the system microprocessor retrieves and stores both the EDAC status word and the data buffer address in an error buffer, as shown in figure 3-22. The EDAC status word indicates the type of error and, for single-bit errors, indicates which MA is at fault. The data buffer address tells which bit of the MA word is bad, simplifying the determination of hard or soft failures. The data buffer address also indicates if the buffer itself has become faulty.

TABLE 3-9. BINARY STATUS WORDS FOR PSEUDO-CYCLIC CODE

DECODING RESULT	DECODER ACTION	BINARY STATUS WORD	ERROR MANAGEMENT ACTION REQUIRED
SINGLE ERROR IN INFORMATION POSITION	ERROR CORRECTED BEFORE OUTPUT OF INFORMATION	0 TO 63	TEST FOR POSSIBLE RECONFIGURATION OF INDICATED CHANNEL AT CURRENT MEMORY ADDRESS
SINGLE ERROR IN CODE POSITION	OUTPUT CORRECT INFORMATION	64 TO 71	
SOME (NOT ALL) OCCURRANCES OF AN ODD NUMBER OF ERRORS ≥ 3	OUTPUT INFORMATION WITHOUT CORRECTION	72 TO 126	TEST CURRENT MEMORY ADDRESS FOR AN ODD NUMBER OF MULTIPLE ERRORS
EVEN NUMBER OF ERRORS		127 TO 254	TEST CURRENT MEMORY ADDRESS FOR AN EVEN NUMBER OF MULTIPLE ERRORS
NO ERRORS	OUTPUT CORRECTION INFORMATION	255	NONE

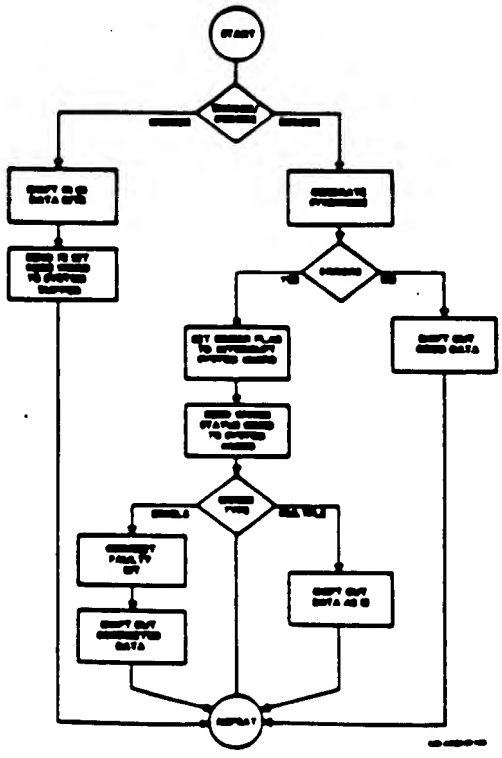


Figure 3-22. Flow Chart of EDAC Operation

As time permits, the system microprocessor transfers the information from its data error buffer to a data error file, which is structured as shown in figure 3-23. The header of this file is the data file number. The entries in this file are subdivided into MA numbers. Under each MA number are the data buffer address where an error for that MA occurred, and the number of errors that occurred at that word. This error file structure allows the compression of data error information to result in simplified diagnostics during built-in-test (BIT).

Multiple bit error information is stored separately since it requires more complicated diagnostics. As multiple bit errors occur, the system microprocessor updates the system status word to indicate an uncorrectable error has occured so the outside world can react properly. The system microprocessor will maintain a count of multiple errors which, with other information dictates the priority of the upcoming BIT.

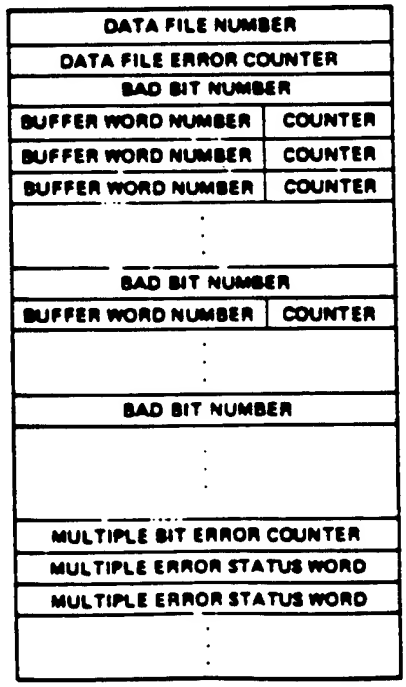


Figure 3-23. Error Status File Structure

3.7.5 System Data Buffer Memory

During normal read/write operations the system buffer acts as the interface between the EDAC and the memory wafers. The system buffer is a pair of 72 x 64 memory arrays which perform a bit transformation on the data passing through. During a write cycle, 72-bit words are sequentially loaded into one of the buffer arrays from the EDAC until its 64-word capacity is reached. Then the data are shifted out, in two 4-bit groups, to the memory arrays. To achieve this the buffer arrays are set up as an array of 4-bit registers arranged in rows of 18 and columns of 64 as may be seen in figure 3-24. After data from the EDAC is loaded into the buffer array by rows, 72 bits at a time, the full buffer array sends data to the memory wafers by columns, 8 bits at a time. An entire column of 8 bits is sent to the mass memory before another column is started.

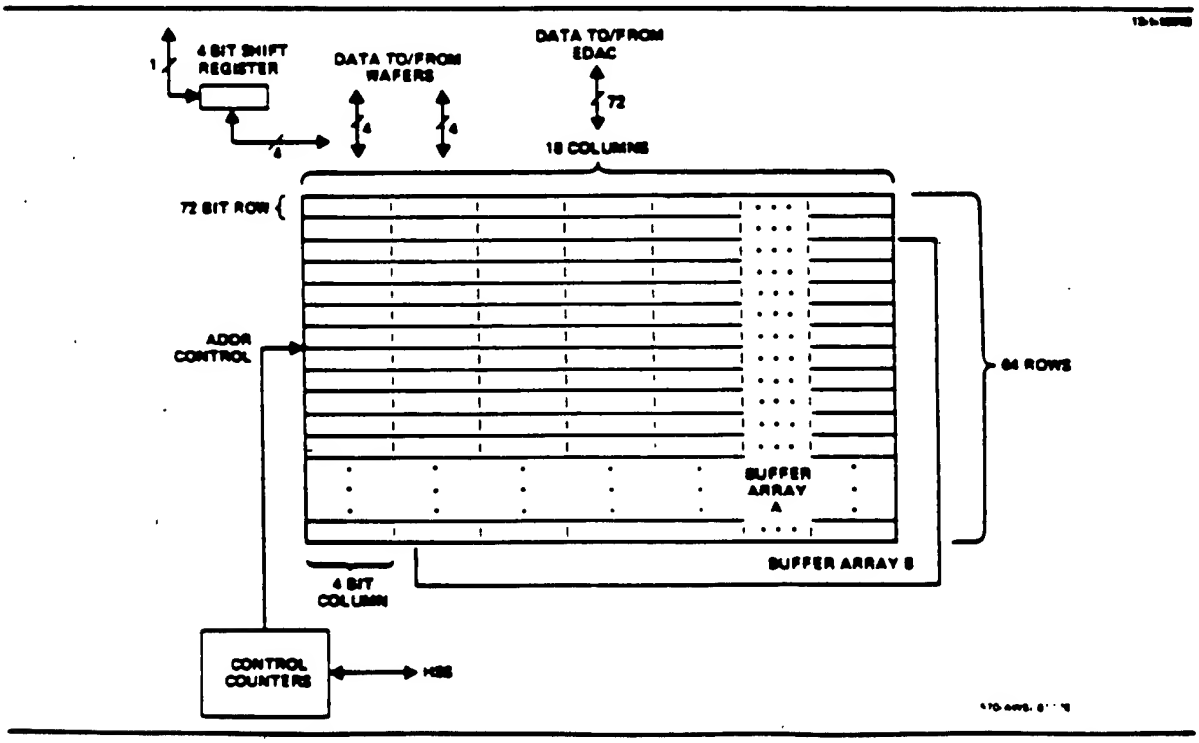


Figure 3-24. System Data Buffer Memory Array

This bit transposition of the data allows each word to be distributed among 72 memory arrays. Then, should one of the arrays be faulty, the EDAC can correct the resulting single-bit error to preserve the integrity of the data. The reconfiguration and testing triggered by the detected error minimizes any multiple-bit errors caused by more than one MA failure.

To maintain the required speed (5 Mb/s) of the memory system, two quadrants utilizing 8 MA's are addressed simultaneously. To facilitate this configuration, two columns of the buffer array are shifted out simultaneously. Since the EDAC cannot wait until the entire buffer array is unloaded before refilling it with new data, the system buffer is arranged as a pair of buffer arrays, A and B. While recently filled buffer A is unloading, the EDAC is filling buffer B with new data. When buffer B is full, buffer A has been emptied and their roles reverse. A signal is sent to the HSS, when a buffer array is filled, to maintain synchronization.

For a read operation the system buffer functions in reverse. Columns of the buffer array are filled with data from the memory wafer. Then row-by-row the retransposed data is sent to the EDAC.

The system buffer is controlled by a set of three 7-bit counters which indicate the row, column, and buffer array currently active. Two counters indicate the column pair and row number for the memory wafers, and the third counter indicates buffer array and row number for the EDAC. These counters also provide the control response signals to the HSS.

3.7.6 Additional EDAC Functions

Besides performing the normal read/write operations, the EDAC also performs the encoding and decoding of quadrant address data for the HSS and of test data for the system microprocessor. The decoding and encoding of all data is accomplished in the same way, except some data are supplied to the HSS and some to the system microprocessor. The HSS supplies mode control information to set up these three operations.

To read or write data in a certain data file, the addresses of the associated memory arrays must be fetched from the wafer. A minimal address fetch time is achieved by storing all eight 64-bit address words in a single quadrant, two words in each of the 4 MA's of the first data block. This only requires two accesses, or approximately 120 microseconds. Since these address data are critical to the memory system operation, they are encoded by the EDAC to guarantee its integrity. To prevent the loss of address information caused by failures, copies of the data are spread throughout the memory. When accessed, the groups of four bits are loaded into the system buffer in column form as regular stored data, but loaded across a row in 4-bit groups until six 72-bit words are formed. Each of these 72-bit words passes through the EDAC for error checking and is then stored in an address buffer in the HSS. The mode controls from the HSS alter the control counter logic to facilitate this format change.

The HSS mode controls can also arrange the system buffer to receive data from a single MA. A 4-bit shift register is used to convert the 1-bit wide data stream into the groups of 4-bit words required by the system buffer format. These 4-bit words are then loaded into the buffer array as was done for the address files. This added capability allows the testing of a single MA.

Since any malfunction in the EDAC results in some form of data loss, the EDAC is tested before any data transfer can take place. The system microprocessor performs the test of the EDAC by supplying known data words in serial form to the encoder inputs. The EDAC then encodes these words and stores them in the system buffer. When all encoding has been completed, the system microprocessor fetches the 72-bit words from the system buffer in 8-bit groups across a row. The encoded word is then checked by the system microprocessor against known results. The decoding and error correction testing of the EDAC is performed similarly, by loading coded test words into the system buffer in 8-bit groups across a row by the system microprocessor. The decoded serial data from the EDAC is then checked, with the resulting error states, by the system microprocessor. The mode controls from the HSS also set up the counter control logic for these operations.

Since the EDAC, including the system buffer and additional control logic, consists of a few custom CMOS/SOS LSI circuits, the system microprocessor would permanently disconnect the faulty chip, as opposed to groups of chips, and lock-in a good spare for a corrected EDAC configuration.

3.8 POWER ESTIMATES

Through size and integrated circuit complexity approximations, power estimates for the various wafer components have been calculated. Since each integrated circuit on the wafer has its decoder powered separately from the rest of the circuit when power is applied to the circuit's bus, decoder power is calculated separately. The power for each type of decoder has been estimated as follows:

CC decoder	15 mW
AC decoder	7.5 mW
MA decoder	7.0 mW

Since in the maximum case there are 11 MA's per X-bus, 77 mW must be allowed for each active X-bus. During normal activities, power is applied to two X-buses in a quadrant, which results in a 140 mW total for the MA decoders. In a quadrant there are a maximum of 30 AC's per Y-bus and only one Y-bus per quadrant. Thus, a 225 mW total for AC decoders in one quadrant, is included. The total decoder overhead is therefore 380 mW per quadrant. To reduce quadrant overhead power during write and erase sequences, the AC and MA decoders have been designed to switch to an inactive state that reduces the power to 3.1 mW per decoder, or 155 mW per quadrant.

When a wafer is powered, the CC decoders are also powered. Since there are 8 CC's on a wafer, 120 mW CC decoder power per wafer is used. Wafers have been grouped in sets of 32 which are powered simultaneously; 3.8 W per wafer group is therefore allowed for CC decoders.

Through comparisons with the existing memory chip, the memory and controller chips have been estimated:

CC 300 mW
AC 100 mW
MA 350 mW

For an active quadrant there are four fully powered MA's, two powered AC's, and one powered CC for 1.9 watts total power per active quadrant. During write and erase sequences the controls assume an inactive state after addressing and require only 100 mW for a CC and 50 mW for an AC. An inactive quadrant then has 1.6 watts total.

The system controller subsections have been estimated by totaling the individual proposed LSI/MSI device power requirements. The three subsystems require:

System Microprocessor	5.0 W
High-Speed Sequencer	3.0 W
Power Kernel	0.5 W
System Controller	8.5 W

The EDAC and system data buffer memory are implemented in custom CMOS/SOS LSI. Their power estimate is based on LSI CMOS circuit information:

EDAC & Buffer	1.0 W

(If different for CMOS/SOS, it will be lower.) Since the EDAC and system buffer are superfluous during erase sequences, they are powered down during these operations.

The total power switches for the wafers and other subsystems must also be included. The total of these has been estimated:

Power Switching	2.0 W

The system peak power estimates are shown in table 3-10 for each sequence: read, write, and erase. The major power variations between sequences are due to the number of quadrants powered. Since writing and erasing the MA's requires more time, the sequences of different quadrants are overlapped. Two quadrants are actively being addressed or loaded with data, while four quadrants are inactively applying an erase or write voltage to their MA's.

TABLE 3-10. WORST CASE PEAK POWER ESTIMATES (IN WATTS)

	READ	WRITE	ERASE
Wafer Overhead	3.8	3.8	3.8
Active Quadrant Overhead	0.76	0.76	0.76
Inactive Quadrant Overhead	-0-	0.62	0.62
Number/Power of Active Quadrants	2/3.8	2/3.8	2/3.8
Number/Power of Inactive Quadrants	-0-	4/6.4	4/6.4
System Controller	8.5	8.5	8.5
EDAC - System Buffer	1.0	1.0	-0-
Power Switching	2.0	2.0	2.0
Total	19.9	26.9	25.9

All power estimates for the single file operations, read, write, and erase, are summarized in table 3-11. A single file operation begins with the fetching of 512 bit file address table; and concludes when the read, write, or erase operation is completed by the 64th word of the 18th quadrant, even though the completion may occur after the last data bit leaves the error detection/correction subsystem. The times for each operation are included in table 3-11.

TABLE 3-11. POWER FIGURES (EXCLUSIVE OF POWER SUPPLY)

	Average	Peak	Total	Time
Read	18.0W	19.9W	2.66×10^{-4}Whr	5.32×10^{-2}s
Write	25.1W	26.9W	3.73×10^{-4}Whr	5.35×10^{-2}s
Erase	23.4W	25.9W	4.13×10^{-5}Whr	6.35×10^{-3}s

There are three types of power values included in table 3-11: average power, peak power, and total power. The peak power is the worst case maximum power used during each of the file operations. These peak power estimates have been updated to include the changes in the maximum number of AC and MA decoders on the X or Y busses.

The average power reflects the small power changes which occur during the addressing of a quadrant and the fluctuations in the number of active and inactive quadrants during the cycle of the file operation. An assumption has been made that the peak power changes at the conclusion of the instruction which causes the power change. This assumption leads to slightly (.1W) pessimistic power figures, but greatly simplifies the calculations.

The average power includes the file address table fetching which uses about 16.3W for 202 microseconds. This fetching is also included in the total file operation times. The operation times given assume that each instruction is completed without any delay due to errors; that is, the instructions are sequenced at their maximum rate.

The total power reflects the entire amount of energy used during a single file operation. This total energy is just the average power used during the entire time of the file operation (average power x time).

The power estimates included in table 3-12 reflect the input power to the system power supply. A 70 percent efficient power supply is assumed; this explains the 43 percent increase in all of the power figures.

TABLE 3-12. POWER FIGURES (INCLUDING POWER SUPPLY)

	Average	Peak	Total
Read	25.7W	27.5W	3.80×10^{-4}Whr
Write	35.9W	38.4W	5.32×10^{-4}Whr
Erase	33.4W	37.0W	5.90×10^{-5}Whr

Section 4
SYSTEM OPERATION

4.0 MEMORY SYSTEM OPERATION

The memory system acts as a secondary storage device for an external system that supplies the basic control signals and commands to govern the memory system operation. The first step of memory system operation is the initialization of the system controller, which is begun when the external system supplies the proper control signals to the power kernel. After initialization is completed, the external system supplies the required system commands to the memory system. The memory system microprocessor receives and interprets the commands into instructions for the high-speed sequencer. The high-speed sequencer then generates the proper sequence of memory wafer commands. Before the high-speed sequencer can issue this sequence of memory wafer commands, it must be supplied the necessary addressing information by the system microprocessor.

4.1 SYSTEM INTERFACE

The system interface provides a connection between the memory system and the external systems. It is divided into three sections: the data interface, the control interface, and the power interface. Each section, as shown in figure 4-1, directly interfaces with a different subsystem. The data interface allows communication with the error detection/correction circuitry, the control interface allows communication with the system microprocessor, and the power interface allows communication with the power kernel.

4.1.1 System Power Interface

The system power interface supplies power to the entire system via the power kernel and control signals to the power kernel. Power is supplied from a single

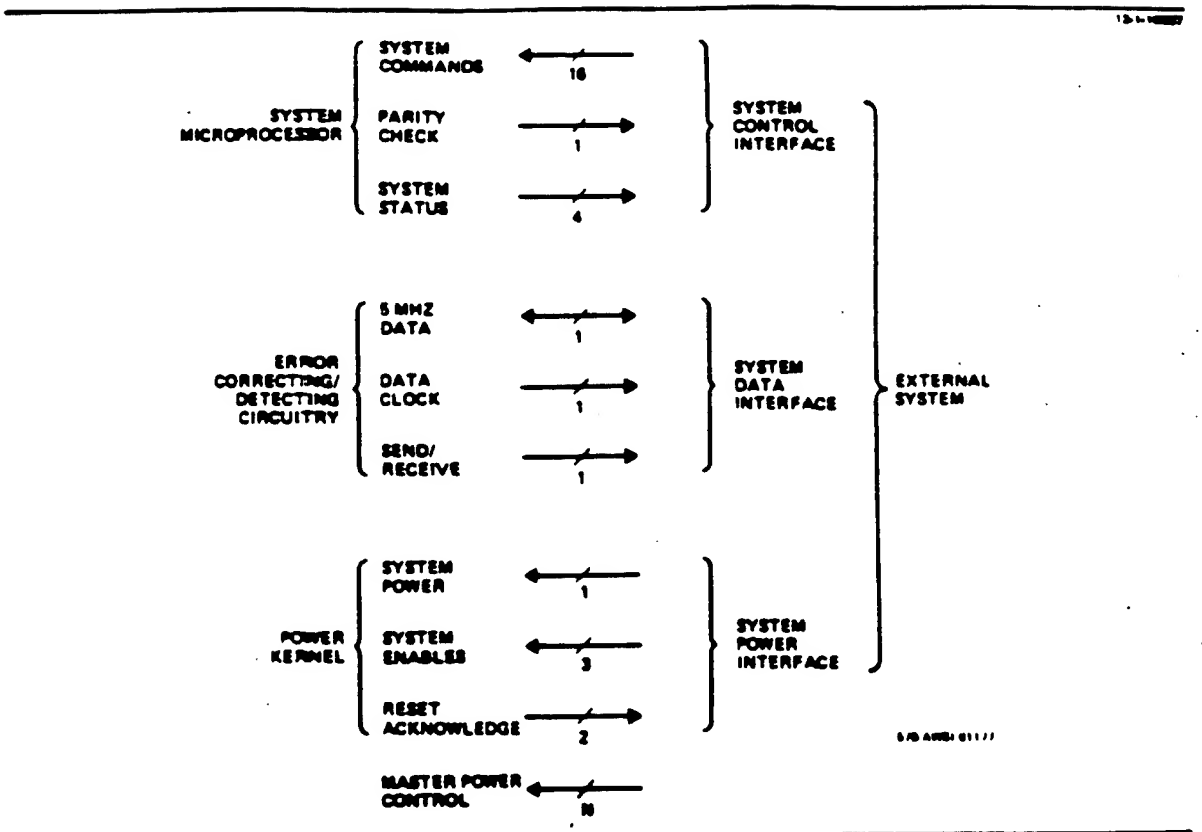


Figure 4-1. System Interface

28-volt supply of the system power supply. The memory system power sypply which is assumed to be 70 percent efficient, provides power to the NMOS/MNOS wafers, plus the TTL 5V.

When the power to the system is initially selected on, only the power kernel becomes active. The system enable signals control exactly those power kernel processors that are activated and start their test and initialization routines as described in section 3.6.1 and 5.1. When the system enable signals activate an operable set of power kernel processors, they respond with an acknowledge signal. Should neither of the system enable signals evoke the proper response, the master control lines are used to effect a brute force reconfiguration of the power kernel.

4.1.2 System Control Interface

Most of the communication between the external system and the memory system is accomplished through the system control interface. The system control interface is composed of the system status and the system command words. The system status word indicates the current state of the memory system and the system command word controls the memory system.

4.1.2.1 System Status Word. The system status word is a 4-bit word which is monitored by the external system. The status word not only indicates the memory system states of busy, ready for a command, or just finishing a command, but also indicates data errors and a need for reconfiguration. The significance of each acceptable status word is shown in table 4-1. The 4 bits which comprise the system status word are known as: busy/$\overline{\text{ready}}$, completion, reconfiguration, and bad file.

The busy/$\overline{\text{ready}}$ bit and the completion bit indicate execution status. The busy/$\overline{\text{ready}}$ bit, bit 0, indicates whether the system is busy currently executing a system command or some self-initiated command, or it is ready to receive a new system command. The completion bit indicates that the memory system has just finished the execution of some process. The completion bit is needed, in addition to the busy/$\overline{\text{ready}}$ bit, to indicate, among other things, that the system is ready to power down. Generally the completion bit and the busy bit can never be set together, these are the only two mutually exclusive status bits. Should both the busy and completion bits be set, an invalid status word or a system microprocessor error is indicated and appropriate external action should be taken.

The reconfiguration bit and the bad file bit indicate error status. The reconfiguration bit, when set alone, indicates that either only single-bit errors have been detected in the data file just read, or that previously detected errors which have not been removed by reconfiguration remain. Generally the reconfiguration bit is set only at the end of a file operation, usually a read, with the completion bit. The bad file bit is immediately set when an

TABLE 4-1. FOUR-BIT SYSTEM STATUS WORD

bit 0 busy/$\overline{\text{ready}}$

bit 1 completion

bit 2 reconfiguration

bit 3 bad file

Status Word	Status Bit Indication	Meaning
0000	ready - no errors	awaiting system command
0001	busy - no errors	executing system command
0010	command completed - no errors	awaiting system command
0011	-	invalid status
0100	ready - need reconfiguration	either single-bit errors were detected in current file or errors still remain from previous file - awaiting system command
0101	busy - reconfiguring	executing reconfiguration and test routines on previously detected errors
0110	command completed - need reconfiguration	detected errors still remain - awaiting system command
0111	-	invalid status
1000	ready - bad file	multiple bit errors were detected in current file - awaiting system command
1001	busy - bad file	multiple bit errors detected in file currently being operated on
1010	command completed - bad file	detected multiple-bit errors still remain - awaiting system command
1011	-	invalid status
1100	ready - severe failure	system microprocessor was unable to resolve severe failure - awaiting system command
1101	busy - severe failure	executing reconfiguration and test routines on detected severe failure
1110	command completed - severe failure resolved	successfully recovered from severe failure - awaiting system command
1111	-	invalid status

uncorrectable multiple-bit error is detected, it indicates that bad data is present in the current data file. Should the bad file bit be set, the reconfiguration bit is not also set at the end of the operation, since a multiple-bit error also indicates the need for reconfiguration. Both bits may be set, however, should the situation arise which requires the memory system to stop executing its command and test and reconfigure some critical subsystem. Such a situation can only be caused by a severe failure. Only a detected malfunction in the high-speed sequencer or the error detection/correction circuitry is classified as a severe failure.

All four bits are evaluated together to ascertain the current situation properly. For example, when the error status bits indicate a severe error, bit 0 showing busy indicates that the system is executing test and reconfiguration routines to remove the fault. A completion bit indicates the fault has been removed and the system command should be reissued. When bit 0 shows the system ready, the fault could not be resolved and system initialization should take place.

4.1.2.2 System Command Word. The six system commands are organized into sixteen bit command words, which are given in figure 4-2. The first three bits of the command word form the operation codes which differentiate the commands. The next twelve bits specify the data file to be used in the single-file operations. The 16th bit is a parity check bit which provides some error detection in the system command interface. Should the parity bit indicate an error, the parity bit check line is made high to indicate the command should be retransmitted.

The single data file operations are the operations basic to the memory system: erase, write, and read. Each of these commands must supply the 12-bit data file address of the file to be operated on. The data files are all 256-kbits (plus 32-kbits of coding), and there are a total of 2048 files.

OPCODES (IN OCTAL)	COMMAND
0	NONE
1	READ
2	WRITE
3	ERASE
4	RECONFIGURE
5	PERIODIC BIT
6	POWER OFF
7	NONE

81178

Figure 4-2. System Commands

The multiple data file operations are more concerned with the memory system as a whole: execute reconfiguration, execute system-wide periodic built-in-test, and prepare to power down. The reconfiguration command is generally performed after all file read operations have been completed. The reconfiguration command then causes the memory system to test and reconfigure the memory wafers, to remove the faults detected by the error detection/correction circuitry. Since specific information about these faults has been stored away, this testing and reconfiguration is rather brief and efficient. In contrast, the system-wide periodic built-in-test covers the entire memory system, which includes all the spare memory arrays and controllers. Because of this, it is time consuming and should only be executed when necessary. Periodic execution of the system-wide test is still necessary both to maintain the efficiency of regular reconfiguration, and to prevent a build-up of faults throughout the system.

Upon completion of either the reconfiguration or periodic built-in-test procedures the memory system outputs data concerning the amount and type of units which were replaced. This data specifies only the largest unit which has been replaced, since, for example, when a row of MA's is replaced, only the fact that a row has been replaced can be determined, and not the number of MA's in that row. A log of all this information for the life of the memory is stored in a special file. The entire reconfiguration log is read out by the memory when the reconfiguration command is received without the reconfiguration status bit being set.

The prepare for power down command allows the system to execute an orderly power down, as required. Upon receiving the power down command, the system executes a finalization routine which, at completion, outputs an error-free status word. Once this status word is set, power can be removed from the system with no adverse effects. The all-zeros and all-ones op codes are not assigned any command to help prevent spurious execution during system power transitions.

4.1.3 System Data Interface

The system data interface provides the data link between the external system and the memory system. In addition to the bidirectional data line, a pair of control lines are also provided in the interface. These data control lines allow proper synchronization of the data transmissions between the systems. The first control line, send/$\overline{\text{receive}}$, gives notice to the external system that the data transmission is about to begin. The 5-MHz clock line begins cycling two clock cycles later, and data should be moved one bit for every clock cycle.

4.2 ADDRESSING STRUCTURE

All addresses presented to the memory system refer to a data file and have no identity with a particular physical location. These are called "data file numbers." As shown in figure 4-3, a data file number is 15 bits plus one parity bit where each 12-bit address refers to a data file of 256 kbits, and the 3-bit

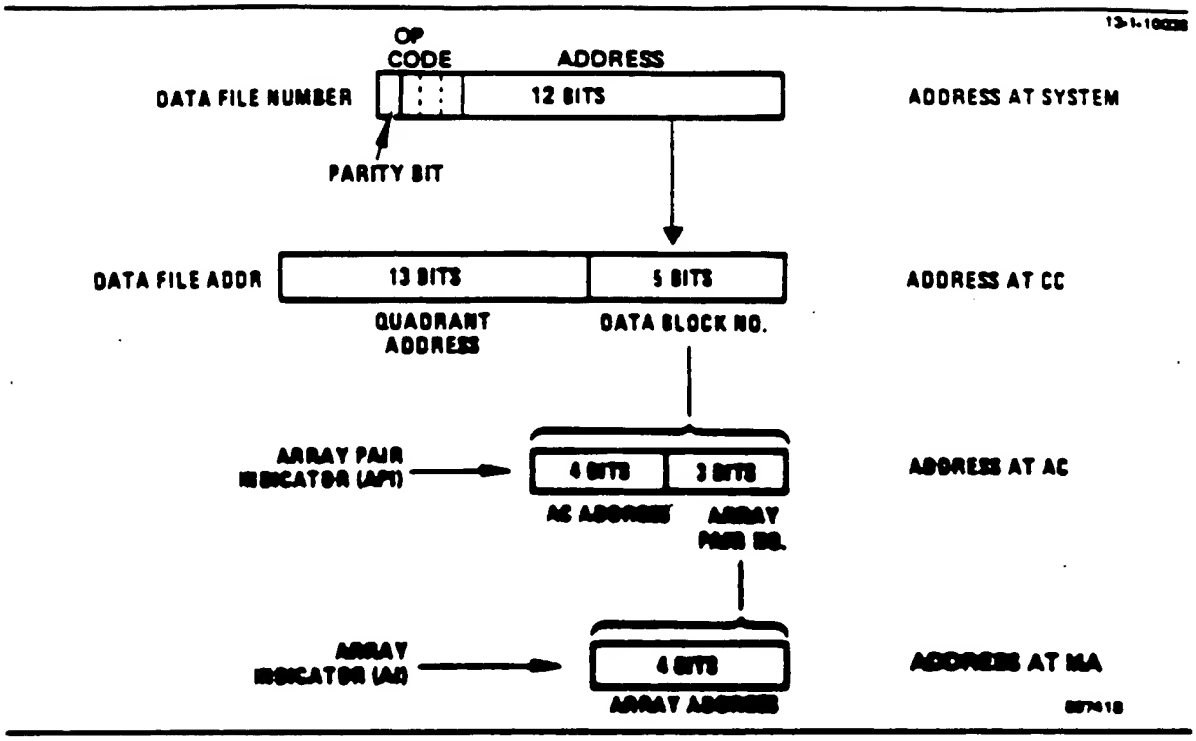


Figure 4-3. Memory Addressing Structure

operation code specifies the memory operation performed on that file. Three levels of translation are performed on the data file number before it arrives at a physical address which references memory arrays. Each translation occurs at an organizational feature of the memory structure.

The 12-bit data file number references a table which translates it into a set of 18 pointers, each referencing a block of 4 MA's in a particular quadrant. Each pointer consists of a 13-bit quadrant address and a data block number; together they form the address information presented to the wafer. The quadrant address selects one quadrant out of 8192. The 5-bit data block number is then presented to the selected quadrant. This serves as an indirect address for two pairs of addresses which identify two different rows in the quadrant; each contains two 4096-bit memory arrays. Each row has a 4-bit address. The data block number also references a 3-bit array pair number which provides an indirect address for the lookup table located in the array controllers. The result is

two 4-bit addresses which give the physical locations of the two referenced memory arrays in this row. With the similar two arrays in the other referenced row, 16 kbits have thus been referenced in one quadrant from the data file address. Similar reference to 17 other quadrants produces a file of 256 kbits of data with their 32k error detection/correction bits. A map of the memory appears in figure 4-4.

Translation of the file number to the addresses of 18 data blocks comprising the data file is accomplished in a four-step sequence as illustrated in figure 4-5. This translation provides the mechanism for a fast and efficient program-controlled access of the required data, as well as several desirable characteristics. Most importantly, the file data blocks are completely relocatable, independent of each other and of the file numbers. Quadrant addresses are completely independent of particular canisters (the wafers are housed in environmentally independent canisters) and their power groups (8 power groups to a canister). The four-step sequence follows:

1. Interpret address of file address table from specific bits in file number.
2. Look up power group in which file address table lies.
3. Access file address table.
4. Access data blocks in sequence using file address table and power group table as required.

Upon request for a file, the file number provides the virtual address for the file address table, which is translated by the quadrant-to-power-group table (QPGT) to a particular physical area in memory. From there the file address table is fetched and stored in active RAM for use as the various data blocks are required. As the entries of the file address table are used to access the various data blocks, the QPGT is used to obtain the physical address for wafer power switching.

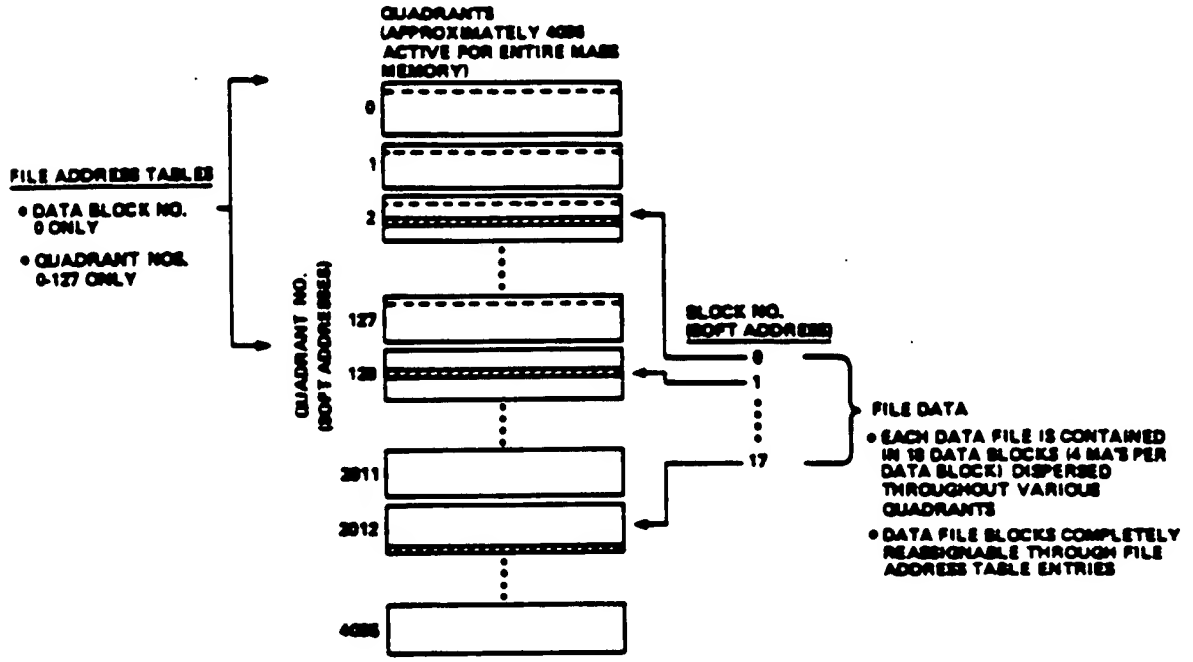


Figure 4-4. Memory Map

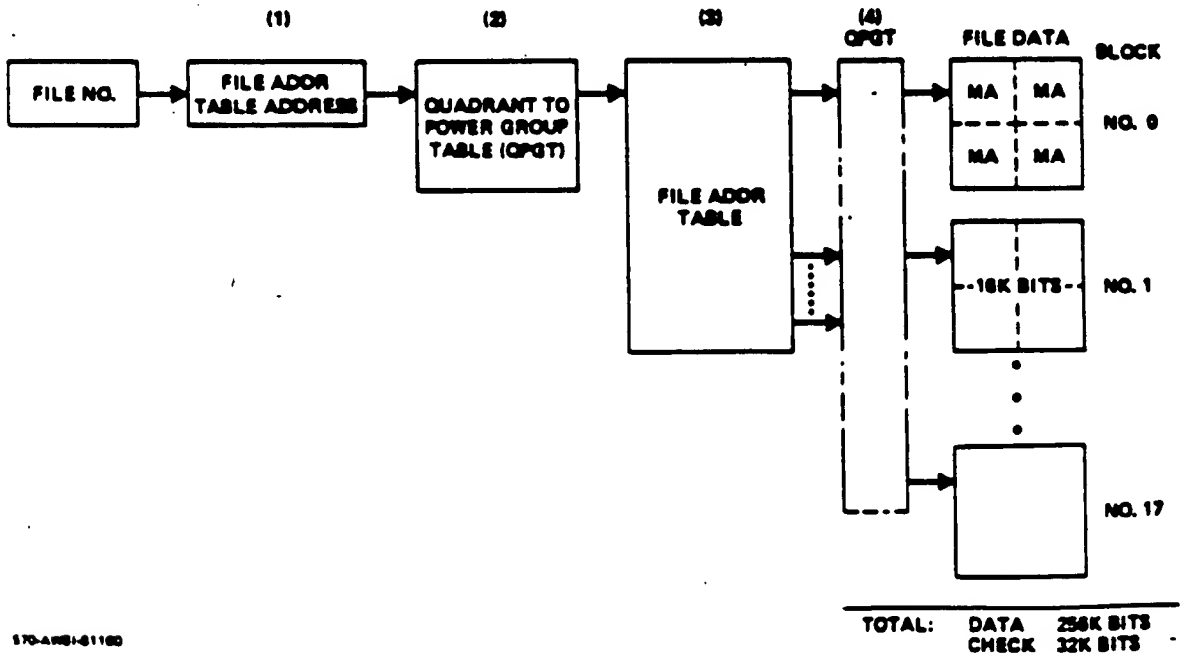


Figure 4-5. File Number to Relocatable File Data

The file address table of the data blocks associated with each file is recorded at the time of configuration in a reserved (relocatable) area of the mass memory. This file address table is configured as shown in figure 4-6a with 18 entries per table referencing the 18 data blocks that compose a data file. Thirty-two file address tables are arrayed per block of 4, 4k-bit MA's (figure 4-6b). The 18 entries of the file address table are packed eight 64-bit MA words with error detection/correction bits included in each word. The table itself is duplicated in another quadrant in a different canister for backup. Two words are unused in the 8-word file address table, with the other six words containing three entries each. Each 18-bit entry in the file address table is composed of a 13-bit quadrant number and a 5-bit block number sufficient to identify the desired data within the mass memory (figure 4-6c).

To locate the file address table associated with a particular file number, the file number is interpreted in its 12-bit format into a 7-bit quadrant address, and a 5-bit table number from most to least significant bit position respectively, as shown in figure 4-7. Since the quadrant address is a virtual address

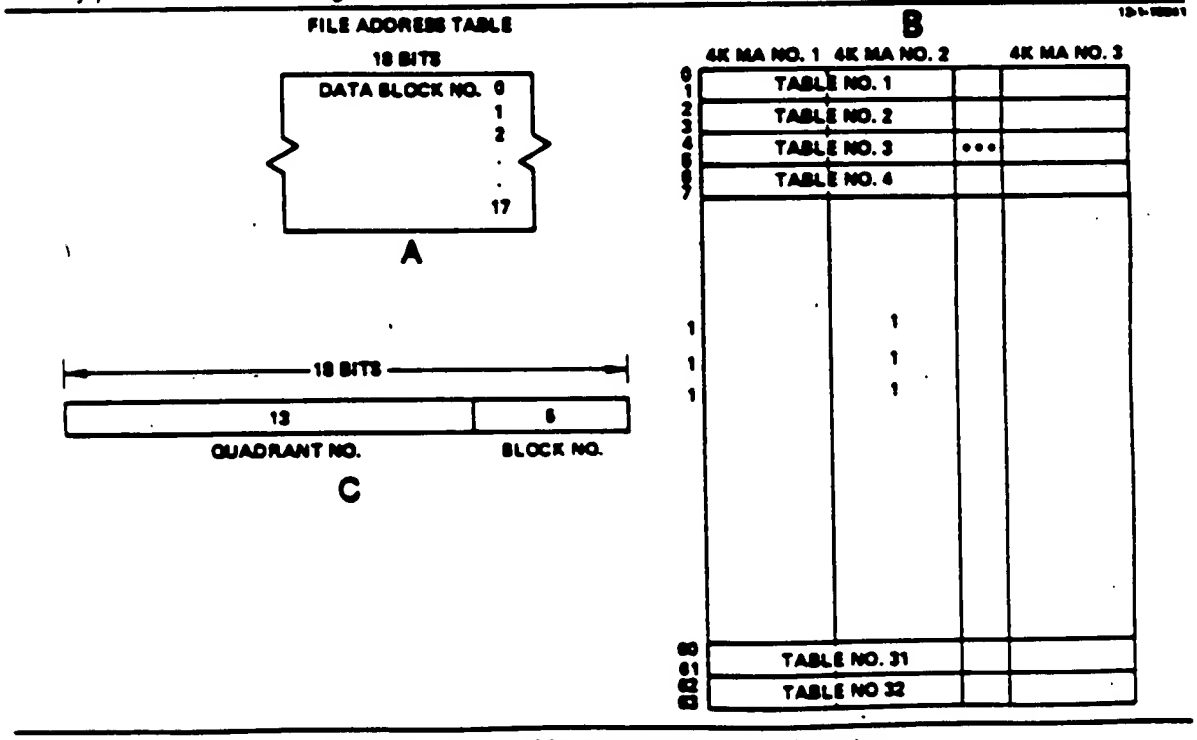
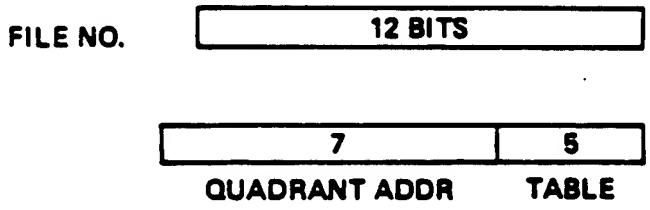


Figure 4-6. File Address Organization

13-1-10042



81162

Figure 4-7. File Number to File Address Table Location

(assignable under program control and subject to change during reconfiguration) its particular real physical area location must be obtained from the quadrant-to-power-group table described in the following section. The sequence to access a requested file is as follows:

a. Using the 7 most significant bits of the file number (the quadrant number) look up the corresponding canister number and power group number in the QPGT.

b. Activate the indicated power group, i.e., switch power to this area, and access the quadrant indicated.

c. Access in block #1 of that quadrant, the table indicated using the five least significant bits of the file number.

d. Load the file address table thus found into RAM and access the 18 data blocks listed in order using the QPGT as required.

4.2.1 Quadrant-to-Power-Group Table

The QPGT supplies the present physical area location of the relocatable quadrant address. As shown in figure 4-8, the table has approximately 5,000 entries of 6 bits each, where each entry corresponds to a particular quadrant address translated to a 3-bit canister number and a 3-bit power group within that canister. The table as stored in RAM is in 8-bit bytes per entry with parity check used each access time. The table is replicated in preassigned nonvolatile bulk memory to the degree to be determined by a reliability analysis.

The quadrant-to-power-group table is updated any time reconfiguration moves a quadrant to a different power group. Of course, any update must be made to all replications of the table. The table is restored in RAM from bulk memory during reinitialization after power turnoff or power failure.

13-1-10063

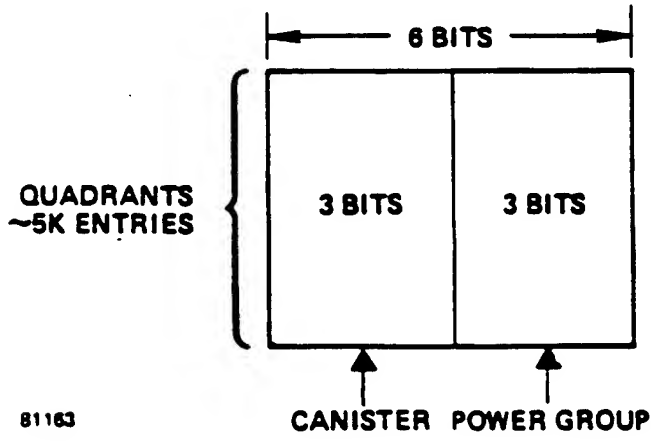


Figure 4-8. Quadrant-To-Power-Group Table (QPGT)

4.2.2 Array Controller Map

Figure 4-9 shows the organization of the array controller map (ACM) located in the CC. The 5-bit data block number references one of 32 word pairs. Each word contains 7 bits: 4 give the AC row address and the other 3 provide the array pair number in that row. Associated with the ACM is a small (16-word x 2-bit) memory which gives the active AC address, indicating which of the 3 candidate AC's is the presently-active one.

The ACM stores not only the actively assigned row and array pair numbers, but also any spare addresses of the quadrant. These are differentiated by storing an all-zero double word between the locations containing the active and spare addresses. The last two locations of the ACM contain a special code word which is used during the built-in-tests of the CC.

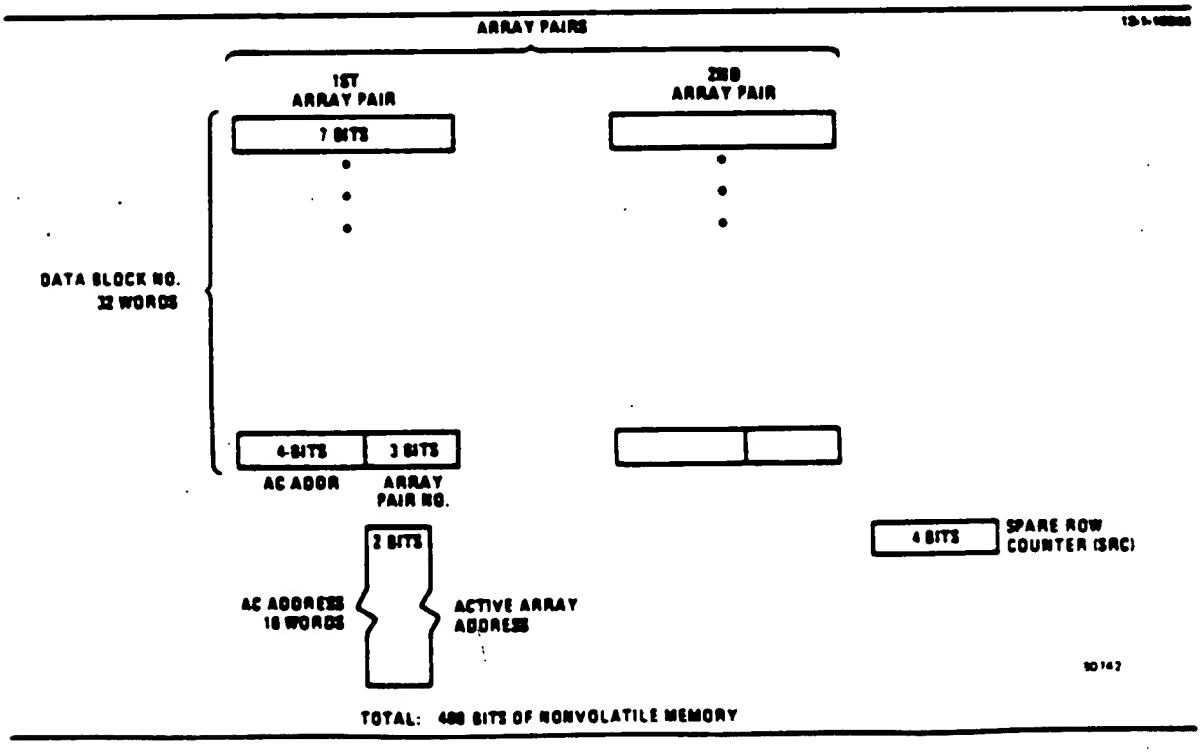


Figure 4-9. Array Controller Map

Also associated with the array controller map is the spare row counter (SRC). The SRC is used during reconfiguration to read and write data of the AOM, and to transmit data between the AOM and the system controller. The system controller reads an AOM location by loading it into the SRC and then counting the number of increments the SRC takes to reach zero.

The array controller map allows for the storage of 512 kbits per quadrant, i.e., 32 entries x 2 array pairs/entry x 2 arrays/array pair x 4 kbits/array. Though current yield figures indicate that only approximately 128k bits per quadrant can be expected, this size of AOM allows the storage of information useful during reconfiguration, and provides for the greater quadrant capacity obtainable with the 4 micrometer, 8 kbit MA without redesign.

4.2.3 Array Map

Figure 4-10 presents the array map (AM) located in the AC. It contains 8 paired entries. Each paired entry is referenced by the 3-bit array pair number. The 5-bit entry is the physical address of the referenced array.

As in the array controller map, the AM stores both active and spare MA addresses separated by an all-zero word pair. The last two locations also store code words used during array controller testing. The spare array counter (SAC) performs the same function for the array map that the SRC did for the AOM.

4.2.4 Address Assignment

Figure 4-11 illustrates the technique of address assignment to the memory arrays in a row with their subsequent address entry into the array map. In the top row, the first array nearest the AC is address "1" with arrays further from the AC having increasing address values. A maximum of 16 array addresses are available; most rows will have considerably fewer MA's. If 16 arrays in this row are found to be operable and 14 assigned active status, 2 will remain for sparing. The arrays assigned to active status are those with addresses of

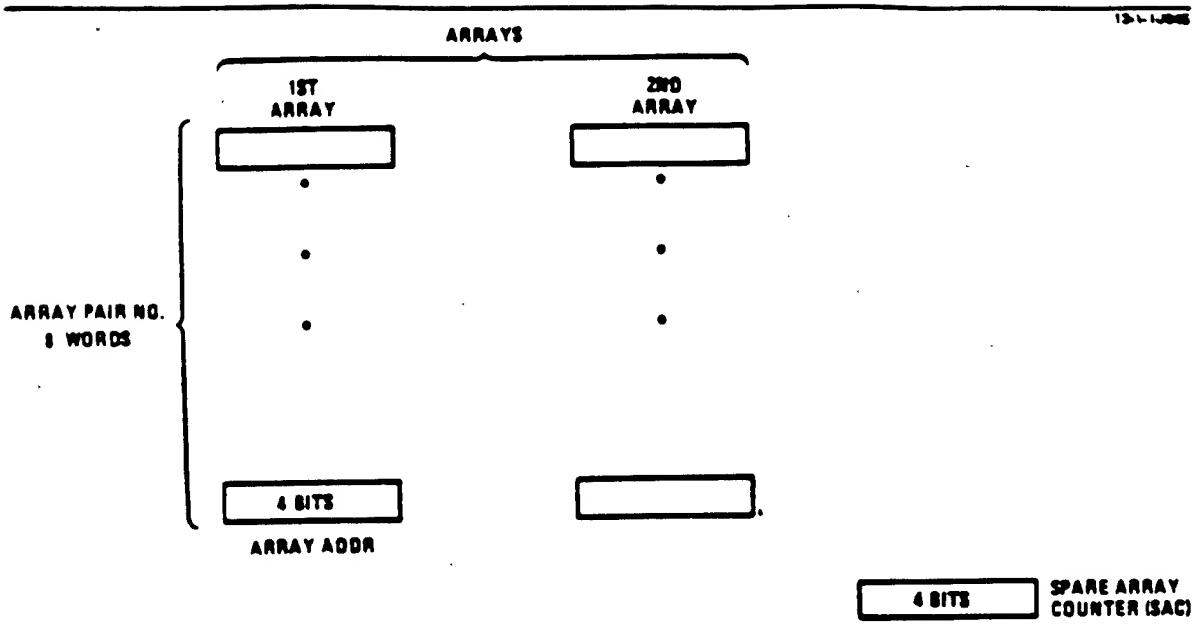


Figure 4-10. Array Map

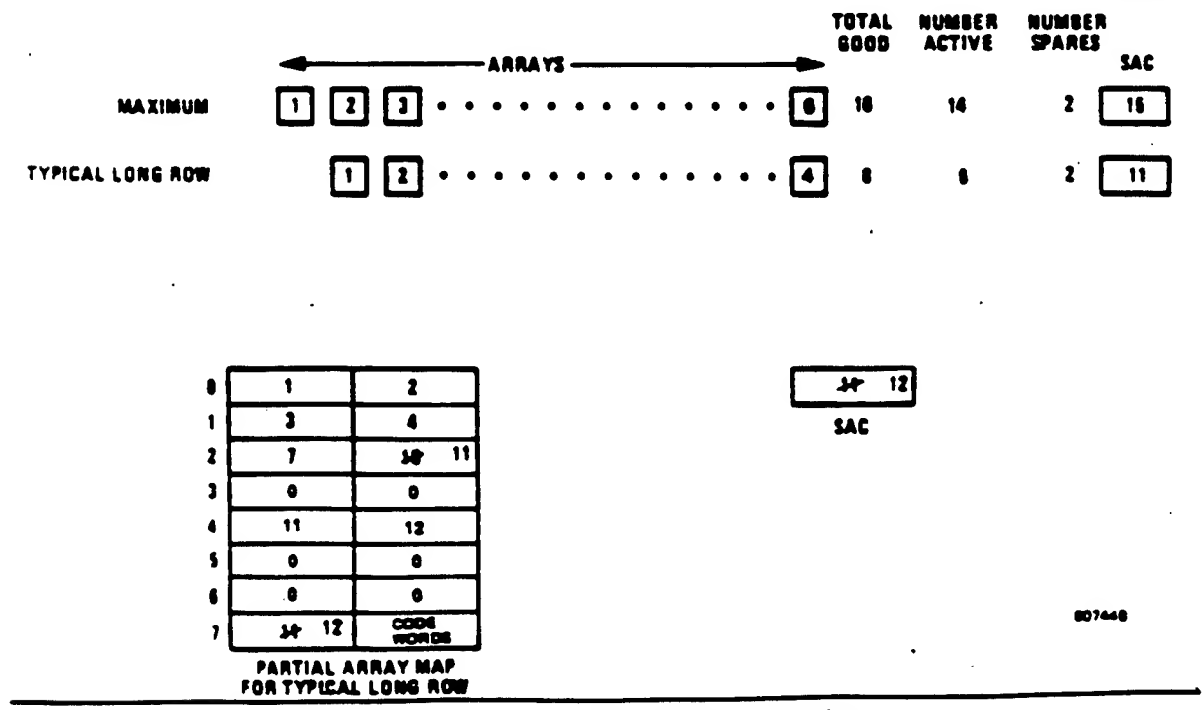


Figure 4-11. Addressing Technique

lower numerical value. The next address in sequence which was not assigned is stored after the all-zero word pair of the array map and loaded into the spare array counter (SAC). When reconfiguration occurs, it will be the first to be used as a spare.

The second row in figure 4-11 presents an example in which 8 operable arrays were found from 14 candidates and 6 were assigned active status; therefore, 2 are in reserve. The array map is built from the operable array information. In this example it is assumed that 3 array pairs were required of this row. Thus, the 0-th entry pair was assigned numbers 1, 2; then 3, 4 and 7, 10, with 11 and 12 entered into word 4 of the array map; word 3 is stored with all zeros to separate the active and spare addresses. (Assume the missing numbers in the sequence of assignment correspond to faulty arrays.)

It is now assumed that built-in-test (BIT) has determined that array 10 in this row has failed and must be replaced. The reconfiguration routine first loads the first half of word 4 into the SAC, after finding it follows the all-zero words, and enters it (11) into the entry where 10 was previously. The substituted array whose address is 11 may or may not be operable. The reconfiguration routine determines its operability. If it is operable, reconfiguration is complete; if it is inoperable, the SAC is loaded with the other spare addresses until the first operable array is addressed. If no operable arrays remain in this row, the data are reassigned to another row.

Because operable arrays are mapped on the wafer, there is a minimum amount of off-wafer overhead. The only temporary external storage required is one 18-bit word per each 16 kbit block of data (18 words total) in the active data file and the QPGT.

4.3 MEMORY OPERATION

Data are written into or read from memory after the proper selection of CC, AC's, and MA's to form a channel. Four memory arrays, in a single quadrant, are accessed in parallel by each channel. To achieve parallel access of eight

MA's total, four data lines, either data bus E or data bus F, are associated with each channel as was shown in figure 3-10. Those data buses are then subdivided into two pairs of data lines, data bus C and data bus D; each of these data line pairs are then split into the single data lines, data bus A and data bus B, associated with the individual MA's.

4.3.1 Read

To perform a read file operation, a pair of channels are opened by addressing two quadrants (CC), two rows (AC) in each quadrant, and two memory arrays in each row. A typical setup sequence, to open a single channel for a read is:

Open first channel
Delay
Select first row
Delay
Select first array
Select second array
Exclusively select second row
Delay
Select third array
Select fourth array
Reselect first row
Delay

The delays in this sequence provide sufficient time for each chip to achieve full power. During each of these delays, similar commands are sent to another quadrant to open the second channel. The channel control signals are used to indicate which channel is to execute a specific command. At the end of the final delay, all eight MA's in the two channels have been addressed and assigned a specific data line.

Upon completion of the addressing of the eight memory arrays, the data is transferred to the system data buffer by the following sequence:

Load word address
Read command
Transfer data
Shift out data

Each of these commands are executed by the eight memory arrays in the two channels in parallel. The shift command is followed by sixty-four 2-MHZ clock cycles which completes the transfer of data over the eight data lines to the system data buffer memory. This entire sequence, as shown in the upper portion of figure 4-12, requires approximately 90 microseconds to execute.

This 90 microsecond sequence is repeated nine times to access a total of 72 MA's in 18 quadrants. (The two additional quadrants, eight MA's, contain the error detection/correction bits.) After the 18 quadrants have been accessed, sixty-four 72-bit words have been assembled in the system data buffer. The system data buffer is filled in this manner a total of 64 times, after which the entire 256k data bits, as well as the 32k error detection/correction bits of a data file have been read out. This read procedure is consistent with the one-bit correctable requirement by having placed each bit of the 72-bit encoded word in a different MA. Thus, with the failure of any single MA, the data is fully recoverable.

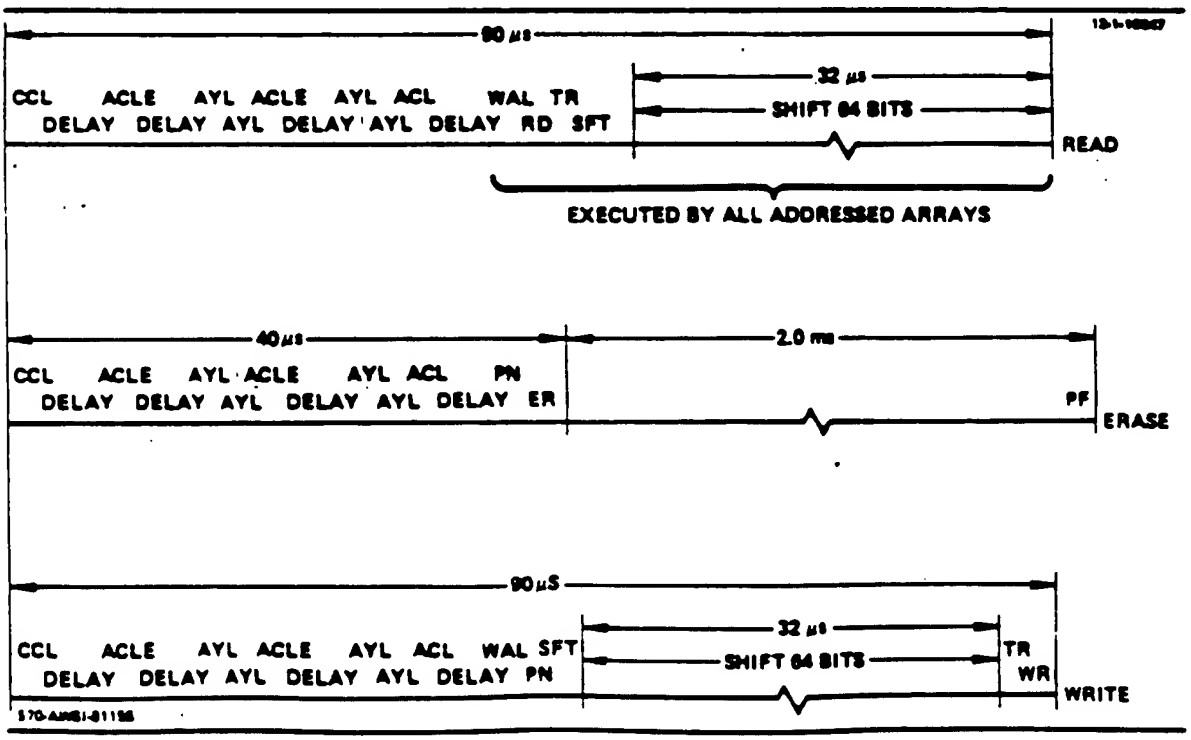


Figure 4-12. Memory Sequencing - Instructions for One of the Two channels

4.3.2 Write

The MA requires a block erase preceding data writing. The erase, as shown in the center of figure 4-12, is a 2-ms period following the proper setup (similar to read setup with the last command "erase" rather than "read"). The erase command can be issued to as many MA's in parallel as the power budget allows. For example, if 0.35 watt is required per MA and 2.8 watts are allotted for on-wafer dissipation, then a data file of 256 kbits would be erased in three 2.13 millisecond periods as shown in figure 4-13. Following erase, data are written into eight MA's in two quadrants at a time preceded by the proper set-up. The write sequence into an array, as it appears in the bottom of figure 4-12, begins much like the read, with a 90 microsecond period when the eight MA's are accessed simultaneously via the two channels. Following the shift in, however, a write period of approximately 180 microseconds is required, as shown in figure 4-14. The setup and data transfer rate is 5.63 Mb/s, but for the full cycle the rate drops to 1.87 Mb/s, which is clearly unacceptable. To overcome this, many quadrants are operated concurrently. Data are loaded into the first

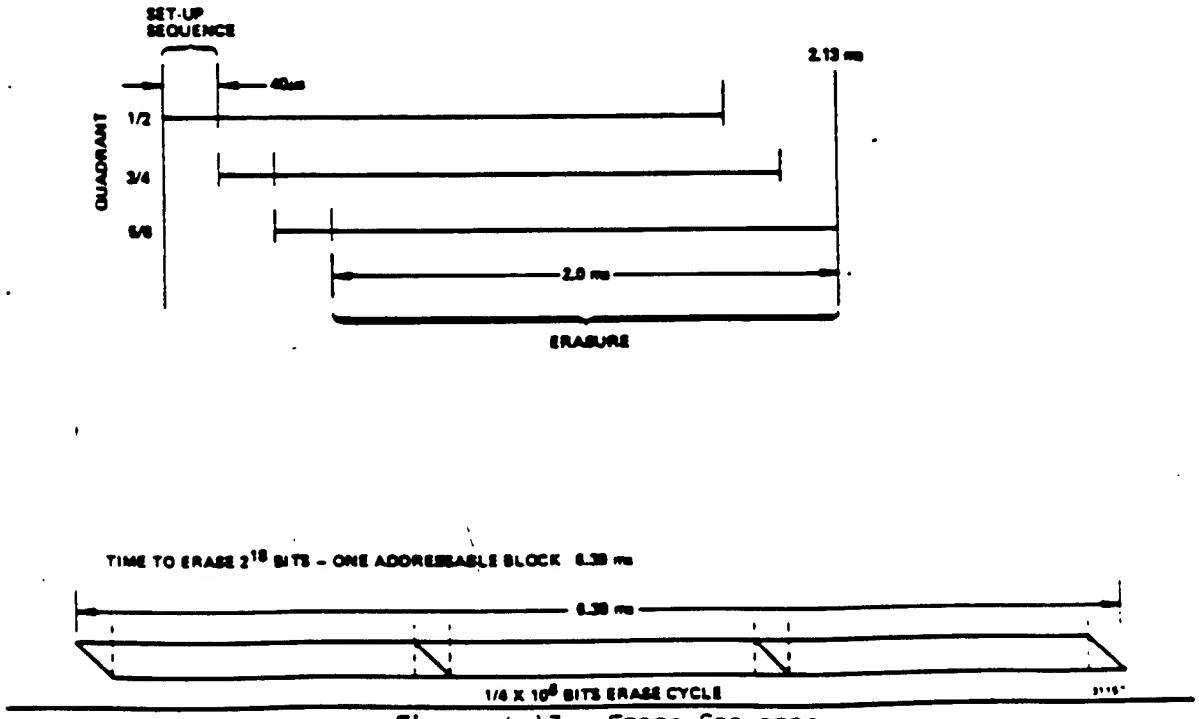


Figure 4-13. Erase Sequence

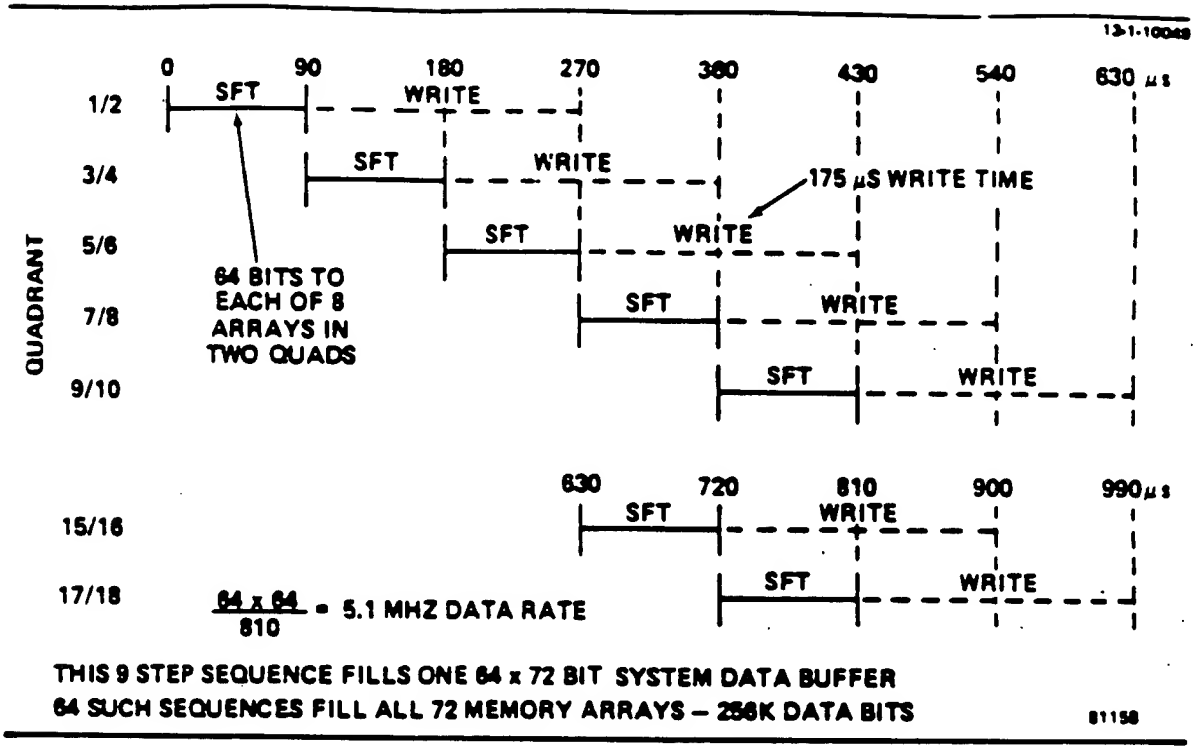


Figure 4-14. Write Sequence

two quadrants and then a 180 microseconds write command is issued. While that quadrant pair's MA's are in the write mode, the system controller loads data into the second quadrant pair, followed by the write command; then, to the third quadrant pair, etc. After a quadrant pair has completed the write sequence, it is powered down and another quadrant pair is accessed to replace it. This procedure continues until all 18 quadrants of the data file have been accessed. In this manner, up to four quadrants are in the write command mode while the two present quadrants are being loaded with data. This sequence (of 64 bits to each of four arrays in a quadrant) for all eighteen quadrants that store the 72-bit words, is executed 64 times. The total write sequence requires 52 ms for all 256k data bits for a data rate of 5.1 Mb/s.

4.3.3 Dual-Channel Operation

The instruction sequences which control the memory wafer execution of each operation are generated by the high-speed sequencer. The set of instructions

which the high-speed sequencer generates were listed in table 3-4. The HSS directs the memory wafer commands to each of the two channels through the use of its channel control (CHC) flip-flops. First, one channel is opened by the channel controller address lock (CCL) command. The setting of the CHC flip-flops during this CCL dictate which channel the channel controller is using. The second channel is opened by addressing the other channel controller, via the CCL, with the CHC flip-flops placed in the opposite setting.

Then the CHC flip flops are alternately set to indicate channel 1 or channel 2 communication, as shown in figure 4-15. The quadrants receive instructions alternately since one is in a required power-up delay mode; this decreases the resultant setup time. Once all eight MA's of the two quadrants have been addressed and are fully powered, both channels are accessed simultaneously to execute the data-related instructions, by supplying the two CHC signals together. The procedure of addressing a second quadrant while the first is waiting for a component to power up, and then simultaneously moving data in/out of both quadrants, achieves the required system data rate with more efficiency.

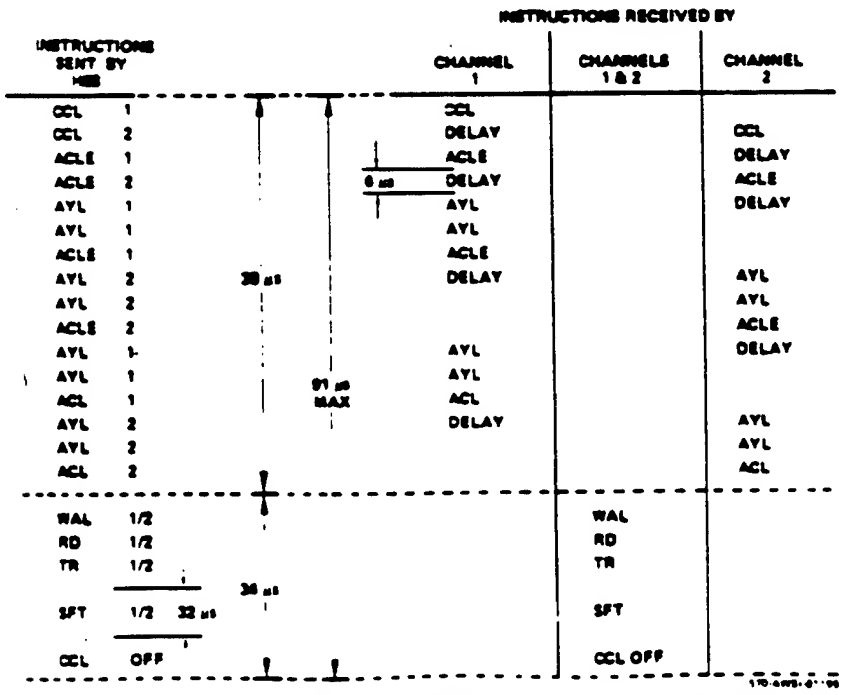


Figure 4-15. Multiplexed Control of Two Channels During A Read Operation

Figure 4-15 shows a read sequence being executed in only 64 microseconds, much less than the 91 microseconds maximum required for the data rate. This excess time allows the HSS and system microprocessor to exert better control over the system.

The wafer instruction sequence for the write and erase operations remains basically the same as the read sequence. The HSS issues the same addressing sequence, multiplexed between the two channels, for all three operations. Only the data sequence, directed to both channels, differs in the three operations as indicated by figure 4-16. The main difference between the write and erase sequences, and the read sequence, is that due to the extended write and erase operations, the addressing and data subsequences overlap the actual erases and writes.

4.4 MEMORY SYSTEM CAPACITY EXPANSION

The memory system capacity can be altered in two increments. The larger increment is 107.4 Mbits of data which constitute a full canister of memory wafers (256 wafers). The smaller increment is the 13.4 Mbits which make up a power group of 32 wafers. There are eight power groups in a canister.

A full canister contains 107.4 Mbits for data, 14.2 Mbits of error code, and 6.4 Mbits of spare memory. A total of 8 canisters can be incorporated into the system, for a total of 1.024 Gbits of memory, before the on-wafer address space of the system is exceeded. Although more than eight canisters can be used in a system, it requires that the power switching become a part of the wafer addressing. Changing the number of canisters has little affect on the speed or power specifications.

The smaller memory capacity increments can be based on power groups. A power group contains 13.4 Mbits of data, 1.8 Mbits of error code, and 0.8 Mbits of spare memory. A memory system based on power group units can either use a different size of canister or partially filled canisters for packaging.

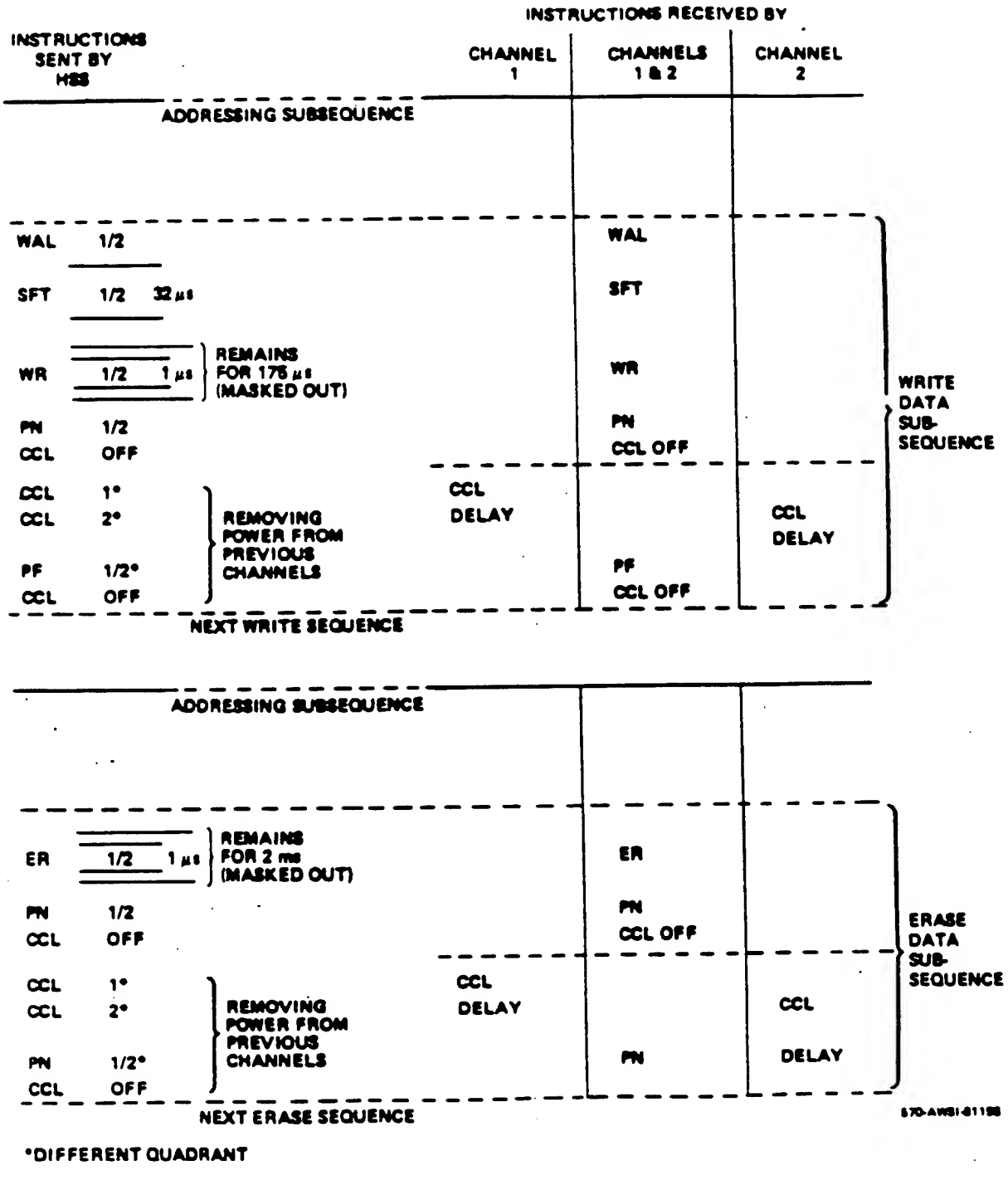


Figure 4-16. Multiplexed Control During Write and Erase Operations

Changing the memory size affects the number of bus drivers, power switches, etc. It does not greatly impact the system controller. The quadrant power group table is the most obviously affected section of the system controller, since it will have more or fewer entries depending on the size of the memory. The size of the RAM required to store the table in the system microprocessor will also change accordingly.

Memory capacities exceeding 1.024 Gbits will require restructuring of the file address tables. Major increases in the memory capacity also affect the memory system interface since more bits must be added to the data file number supplied to the system. Reconfiguration is also affected by memory capacity changes, since more memory must be searched to find a spare during major reconfiguration to replace entire quadrants.

The smallest amount of memory which would make a reasonable system is five wafers. This is a 2.1 Mbit system. Five wafers are necessary since 18 quadrants are required for a single 256 kbit data file. However, with memory systems of much less than 107 Mbits, one canister, the system controller begins to show an overkill of capability.

4.5 MEMORY SYSTEM SPEED ENHANCEMENT

The speed of the memory can be increased by operating additional quadrants in parallel. The two major effects of increasing the speed are the system power requirements and the modification of the system controller software. Minor hardware modification is necessary at the interface to the memory wafers to multiplex and control the flow of data from multiple ports. Each of these ports would correspond to a separate active Z-bus. Each is connected to a different canister to operate separate pairs of channels. The memory system could then operate at an additional 5 Mb/s for each added port.

The addition of each memory wafer port would result in approximately 12.5W of additional peak power (9.2W average power) for a read operation and approximately 23.4W additional peak power (20.7W average power) for write operation.

Thus, a modified memory system operating at 15Mb/s would require approximately 86W peak power (77.8W average power) during a write operation.

For memory system operation much above 15 Mb/s, the speed capability of the system controller and error detection/correction circuitry (EDAC) would need to be increased. First, the speed capability of the CMOS/SOS EDAC would be surpassed, and bipolar components such as ECL, would be used. This would result in a power increase of about 5 to 7 watts. Shortly thereafter, the speed capability of the high-speed sequencer would be exceeded and would have to be modified into a "multi-sequencer." The multi-sequencer version of the high-speed sequencer would require an additional Advanced Micro Devices 2910 microsequencer and faster memory for a power increase of approximately 4 to 6 watts. In any case, the maximum speed of operation of such a modified memory system is estimated to be 100 Mb/s using 500W of peak power during write operations. Other organizational approaches should be investigated for significantly higher speed systems. However, system design cannot overcome the basic speed-power properties of electronic systems.

Section 5
SYSTEM CONTROLLER SOFTWARE ROUTINES

5.0 SYSTEM SOFTWARE DESCRIPTION

The relationships between the main routines of the system controller components are shown in the block flow diagram of figure 5-1. The routines shown on the same horizontal row operate concurrently. The three columns indicate the location of the routine: power kernel, system microprocessor, or high-speed sequencer.

The power kernel routine is executed, almost continuously, by each of the power kernel processors simultaneously. This routine checks each of the power kernel processors and the system microprocessor CPU. When the system microprocessor is to be tested, the power kernel processors issue a restart interrupt which initiates its initialization routine.

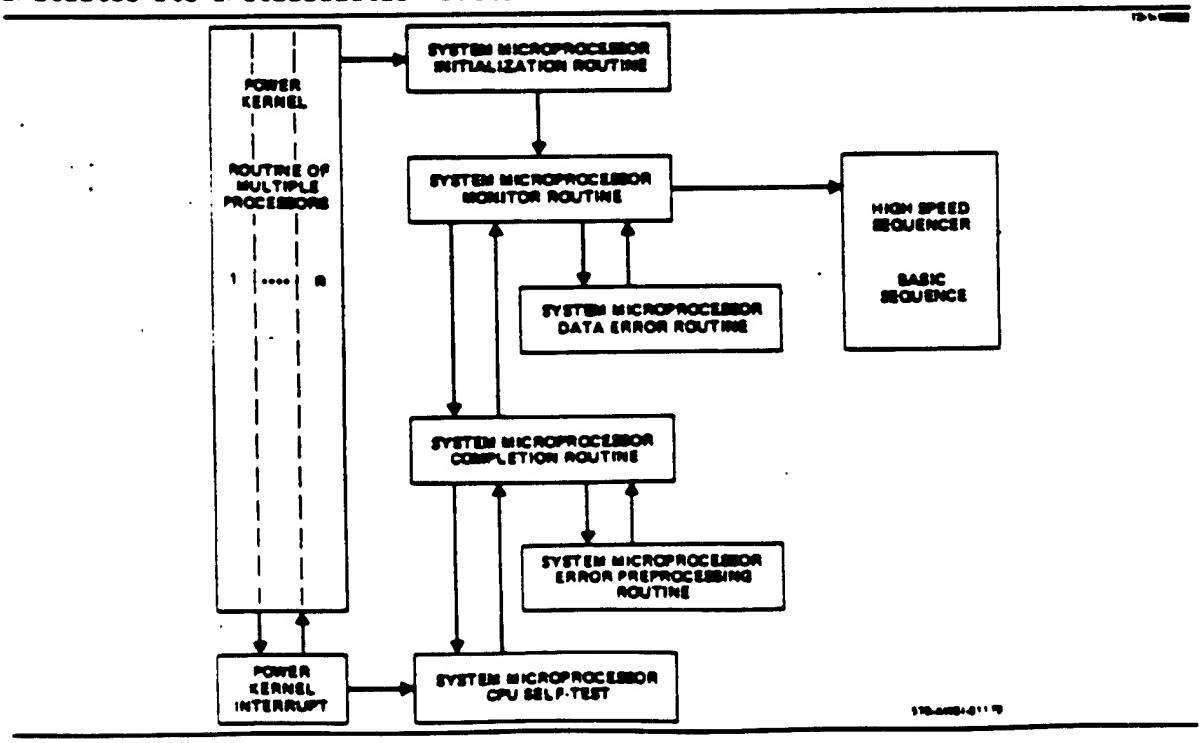


Figure 5-1. Block Flow Diagram Showing Relationships Between Routines of System Software

The initialization routine tests and initializes the system microprocessor, the high-speed sequencer (HSS), and the error detection/correction circuitry (EDAC). After all tests are passed, the initialization routine accepts the first system command and passes control to the monitor routine.

The system microprocessor monitor routine starts the high-speed sequencer generating its basic sequence, as defined by the system command. The monitor routine then checks the progress of the HSS through its basic sequence, and watches for error information from the EDAC. Once the HSS has completed its sequence, it generates an interrupt to the system microprocessor, which causes control to pass from the monitor routine to the completion routine.

The system microprocessor completion routine first checks to see if the system command has been completed, or if another cycle of the HSS sequence is required. If the system command has not been completed, control is returned to the monitor routine. Otherwise, an interrupt is sent to the power kernel causing the power kernel processors to monitor and control a system microprocessor CPU test. Should the CPU fail its test, system initialization is restarted. When the CPU has passed the test, the system microprocessor accepts the next system command and passes control to the monitor routine.

During read operations, the EDAC issues interrupts to the system microprocessor monitor routine whenever it detects a data error. Upon receiving this interrupt, the monitor routine passes control to the system microprocessor data error routine. The data error routine reads the EDAC error status and stores it in an error file. The data error routine then returns to the monitor.

After data errors have been detected, the completion routine will request to pass control to the system microprocessor error preprocessing routine by setting the system status reconfiguration flag. The system reconfiguration command causes the error preprocessing routine to examine the error status information in the error file. Depending on the information in the error file, the error preprocessing routine selects the proper built-in test to isolate the faults which have appeared in the memory arrays and controllers. Once all the

faults have been removed, the error preprocessing routine returns control to the completion routine for the CPU test.

5.1 POWER KERNEL: ROUTINE OF PROCESSORS

The function of a power kernel processor, $processor_i$, is to perform and monitor tests of itself, the other power kernel processors, and the system microprocesssor CPU. The execution of these tests form the main routine of the power kernel processor as shown in the flow chart, figure 5-2.

Immediately upon receiving system power, $processor_i$ begins execution of a self-test routine. This routine both initializes $processor_i$ and fully exercises it. The result of this self test is a number which $processor_i$ outputs to the other procesors of the power kernel. $Processor_i$ compares the results received. If at least one of the results from the other processors matches $processor_i$'s result, then $processor_i$ votes to leave $processor_i$ powered. If the majority of other processors also vote to leave on $processor_i$, the next processor, $processor_{i+1}$, will have its results voted upon. Should a majority of processors disagree with $processor_i$, then $processor_i$ is powered down.

When $processor_i$ is powered down, as a result of this voting procedure, its replacement is powered up. Then the self-test and voting procedure is restarted by all the power kernel processors.

After each of the power kernel processors has had its self-test results successfully checked and passed, a test of the system microprocessor CPU is begun. The status of the CPU is read and the currently active system microprocessor is found. $Processor_i$ then issues a power-on signal to the CPU which is voted on with the rest of the power kernel processor outputs. Once power is applied, $processor_i$ issues a reset interrupt to the system microprocessor CPU, which is voted upon. This causes initialization of the system microprocessor and the start of a CPU self-test. $Processor_i$ then waits for the results of the system microprocessor self-test. Should these results not be received within a given

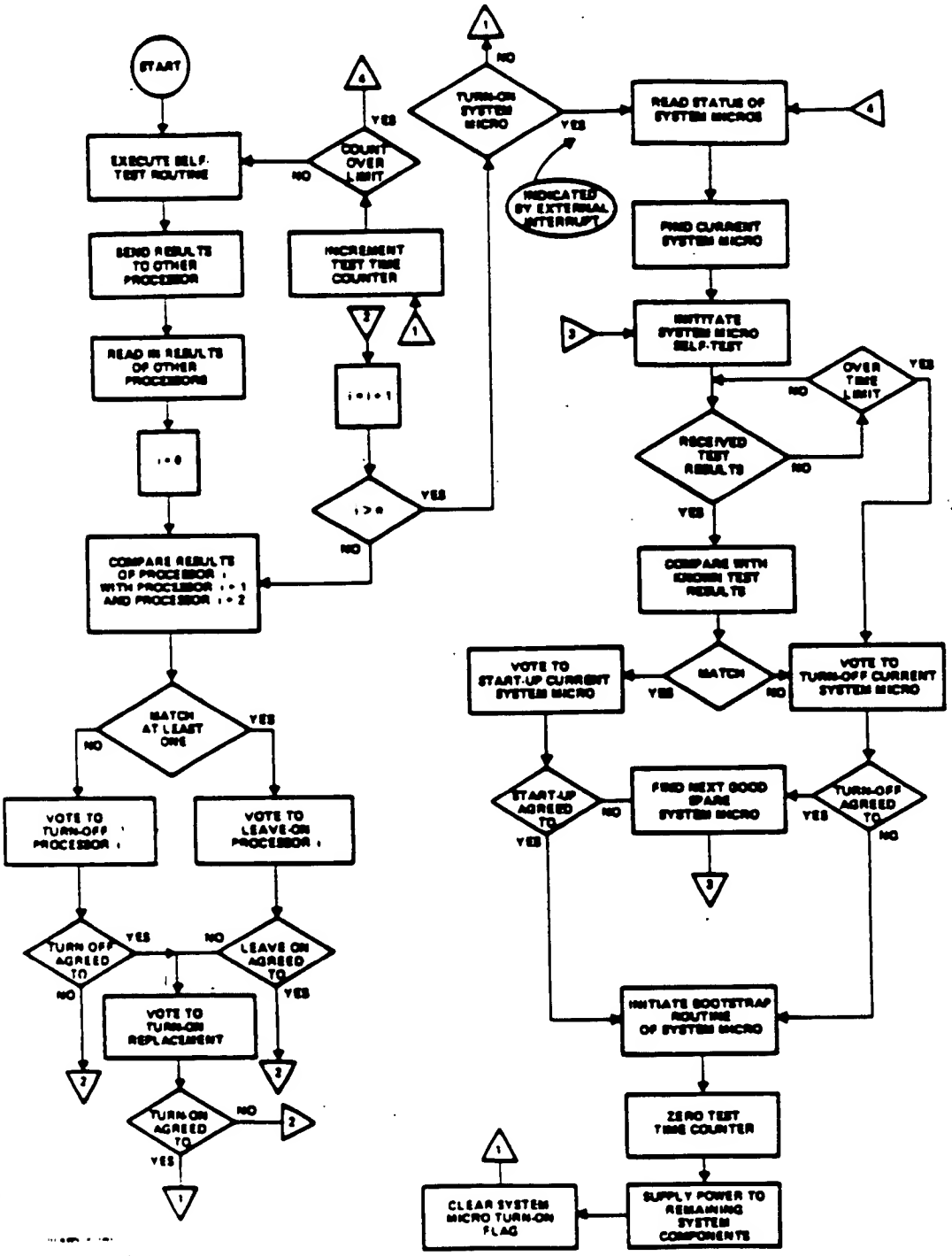


Figure 5-2. Power Kernel Routine of Processor$_i$

time limit, processor$_i$ votes to turn off this current CPU. When the test results are received in time, processor$_i$ compares these with the known correct results of the test. If a majority of the power processors agree on the positive results of the test, then the bootstrap of the current system microprocessor is started. Should the majority of the power processsors agree the current system microprocessor CPU fails the test, it is powered down and the spare CPU next in line is powered up and the test procedure resumed.

Once the power processors agree that the system microprocessor has passed the CPU test, the CPU must be retested periodically to prevent a malfunction from occurring without being detected. The system microprocessor initiates this test between file operations; however, since a file operation may never end if the system microprocessor fails, the power kernel performs a time-out to detect this failure.

After initiating the system bootstrap, processor$_i$ will zero a counter. Between each power kernel self-test this counter is incremented and tested. Should the counter of processor$_i$ reach the allowed maximum, processor$_i$ will begin the CPU test routine by setting the system microprocessor turn-on flag. Initialization and test of the system microprocessor CPU then proceeds as during the initial power up. This system microprocessor turn-on flag can also be set from outside the memory system, should the outside world detect a system microprocessor failure first.

5.2 POWER KERNEL: INTERRUPT ROUTINE

After completing a file operation, the system microprocessor issues an interrupt to the processors of the power kernel. This interrupt notifies the power kernel that an operation has been completed, and that the system microprocessor now has time for a short self test. Upon receiving this interrupt, the power kernel processors stop executing the main routine given in figure 5-2, and begin execution of the power kernel interrupt routine, flow charted in figure 5-3.

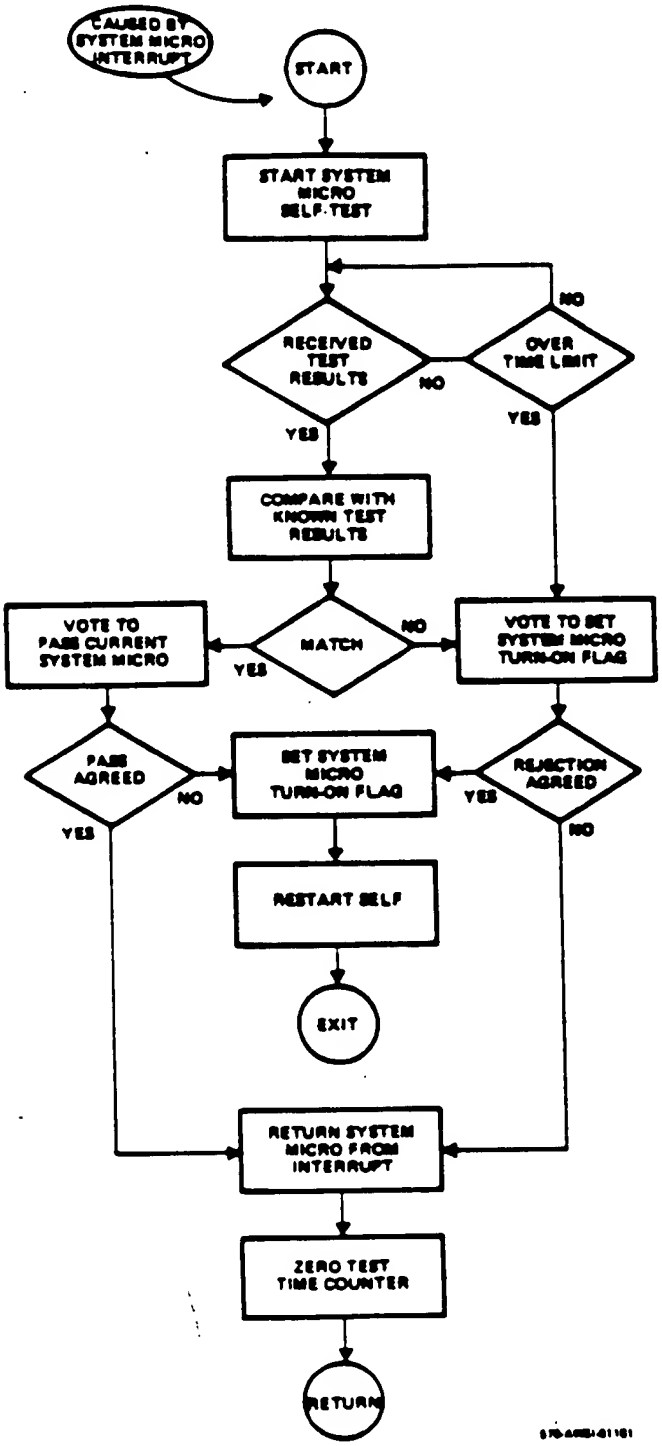


Figure 5-3. Power Kernel Interrupt Routine

The interrupt routine is similar to the system microprocessor CPU test of the basic processor routine. Should $processor_i$ determine the system microprocessor has passed the CPU test, $processor_i$ will vote to pass the CPU. If a majority agree that the CPU passes, then normal operation of both the power kernel and the system microprocessor continues. However, if a majority agree that the CPU has failed its test, then the system microprocessor turn-on flag is set and the full system microprocessor self test is executed by the power kernel as in the main $processor_i$ routine.

5.3 SYSTEM MICROPROCESSOR: INITIALIZATION ROUTINE

The system microprocessor is powered up by the power kernel. Initially upon receiving power, only the CPU and a small section of ROM are active. The CPU then executes a self test routine which resides in the active ROM. After completing this test, the power kernel checks the results. It then either enables the remainder of the system microprocessor initialization routine, or powers up another CPU module. Power is received by the rest of the system after the CPU passes its self test.

Once the power kernel decides the CPU is properly functioning, the CPU will continue testing the rest of the system microprocessor to bootstrap itself to a fully functional level, as shown in figure 5-4. First the microprocessor CPU will check out the rest of its memory. Should the CPU decide that the memory is not properly functioning, it will switch in a spare memory module. After finding a functional memory module, the CPU tests its I/O controller and I/O sections. If the CPU decides that the I/O section is not fully operational, it then switches in a spare I/O module. Once a properly functioning unit is found, the system microprocessor configuration is complete.

The system microprocessor next tests the high-speed sequencer. The system microprocessor enables the HSS to execute a self test routine. As this routine progresses, the system microprocessor monitors the progress of the HSS by reading the HSS-system microprocessor status word. If the HSS achieves its different states in the proper order, the system microprocessor decides the HSS

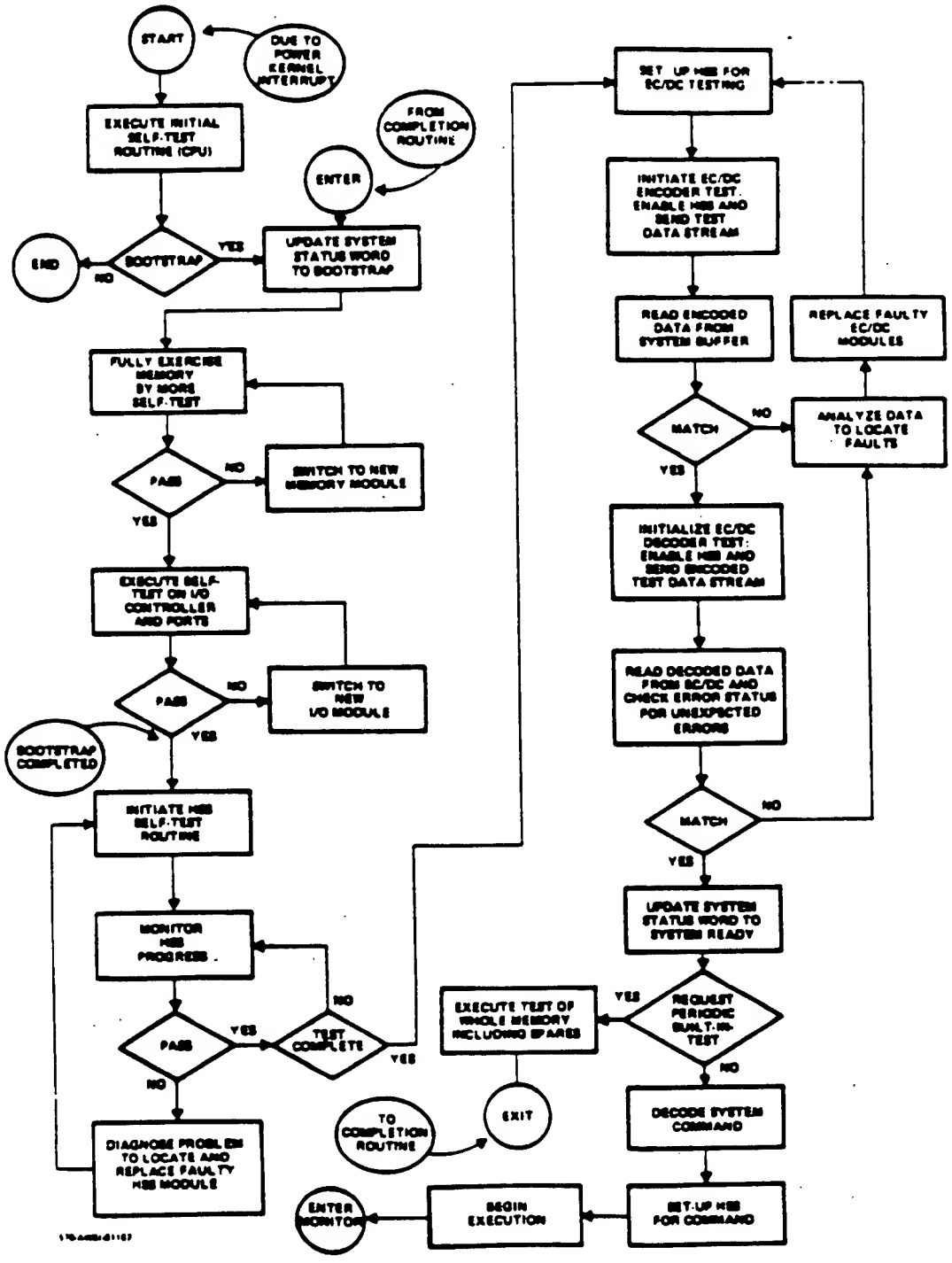


Figure 5-4. System Microprocessor Initialization Routine

is fault free. However, should the HSS achieve an improper state, the system microprocessor analyzes the test state sequence to deduce which HSS module is faulty. The system microprocessor then causes the faulty HSS subsection to be replaced by a new module, and the HSS self test is restarted.

Finally, the error detecting/correcting circuitry is tested. The EDAC is placed in the encoder test mode by the HSS to set up the proper data paths. The system microprocessor then uses these test data paths to supply the EDAC with a known sequence of data. When this data, now encoded, is returned to the system microprocessor, the system microprocessor checks the encoding of the data with proper encoding for the data. If an error occurs, the system microprocessor isolates the faulty EDAC module which caused the failure, and replaces it with a good spare. The encoder test is then repeated. Once the EDAC passes the encoder test, the HSS switches the EDAC to the decoder test mode. The system microprocessor then sends encoded data, which contains some known errors, through the EDAC. Both the output of the EDAC and its error status word are monitored by the system microprocessor. Should an unexpected error be indicated by the error status word, or an expected error not show up in the error status word, a fault is indicated. When a fault is found, the system microprocessor analyzes both the decoded data output and the error status information to identify the location of the fault. Any fault found by the decoder test will be resolved by switching in a spare EDAC module to replace the malfunctioning section. After rectifying a decoder fault, the system microprocessor restarts the test of the EDAC encoder.

Once the EDAC passes the system microprocessor's testing, the system microprocessor refreshes the address tables stored on the wafer. Then the system microprocessor updates the system status to show the memory system has finished initialization. The system microprocessor then requests to perform a test of all the memory wafers, known as the periodic built-in test. If permission is granted, the system microprocessor executes the test, and then enter the system microprocessor completion routine. Permission is not granted for the periodic built-in test when there is another command for the memory system to execute. This command is decoded and the HSS is initialized to execute its part of the

command. The system microprocessor then begins execution of the system command by entering the system microprocessor monitor routine.

5.4 SYSTEM MICROPROCESSOR: MONITOR ROUTINE

While the high-speed sequencer is issuing instructions to the wafer and the error detecting/correcting circuitry is encoding/decoding the data, the system microprocessor monitors the functioning of each to ensure they are executing the proper routines. The monitor routine flow chart shown in figure 5-5 illustrates the method of accomplishment.

A system command is initiated by enabling the HSS, which has been previously set up to issue the sequence of wafer instructions for that system command. The system microprocessor interrupt logic is then enabled so the microprocessor can receive any critical information from the rest of the memory system: wafer addressing errors, EDAC data errors, HSS completion errors, etc.

The system microprocessor monitors the progress of the HSS by reading the HSS system microprocessor status word. The system microprocessor then decodes this status word into a state number, for the current system command. This new state number is checked by looking in the HSS state table, for the current system command, to see if it is a legal state number. If the new state number is legal, it is loaded into the current state counter. If the new state number is illegal, the state number calculations are repeated to ensure that the system microprocessor is not at fault. Should the new state number still be illegitimate, the HSS is disabled and the system status word is changed to indicate a system halt due to a severe failure. An HSS diagnostic is then executed.

Once the system microprocessor has concluded that the HSS has properly progressed to a new state, it checks to see if a new system command has been issued. If no new system command has been issued, the system microprocessor repeats its check of the HSS state. Should a new system command occur, the

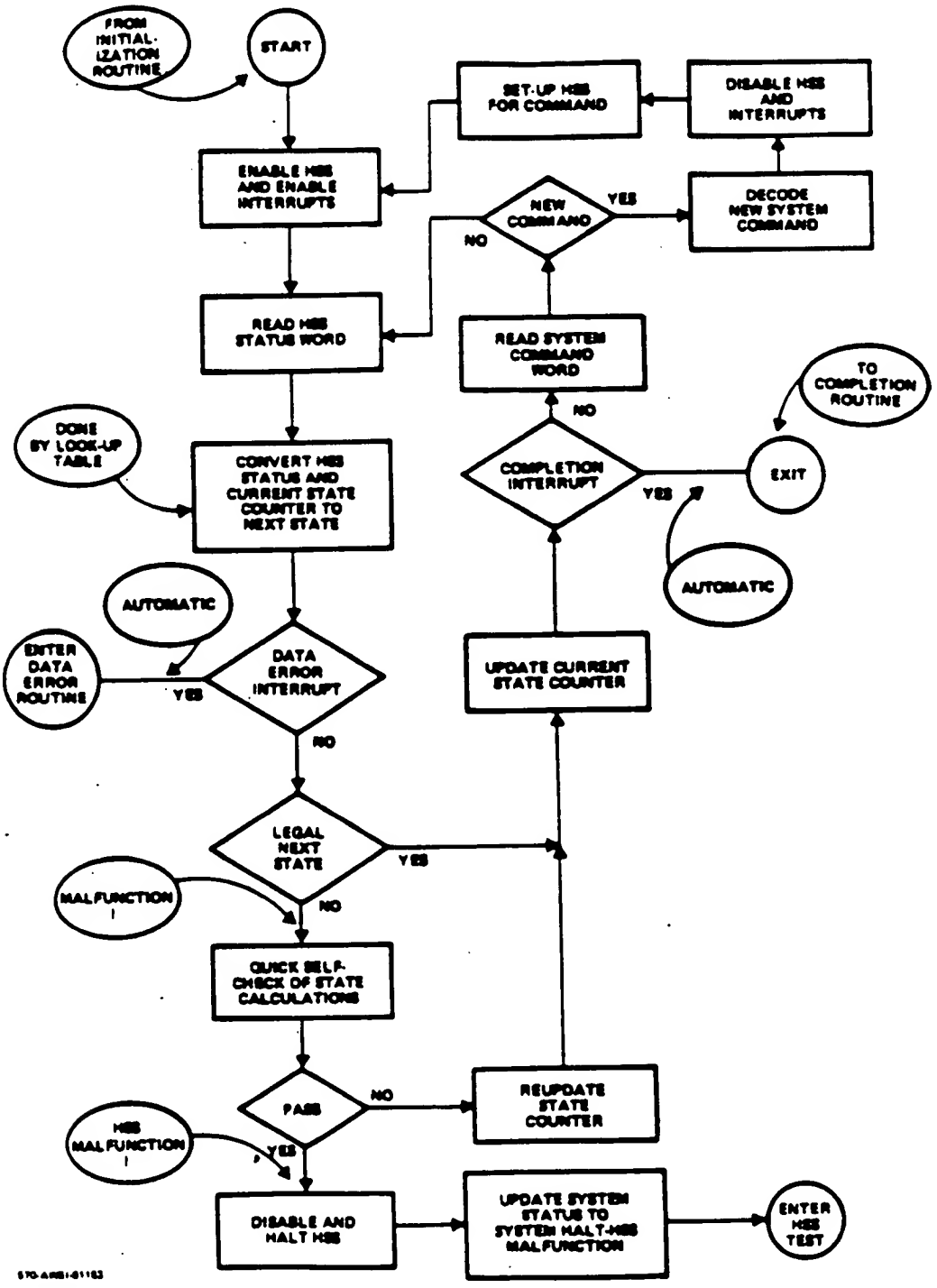


Figure 5-5. System Microprocessor Monitor Routine

LEX15539

system microprocessor halts the HSS and sets it up to issue the new sequence of wafer instructions. Then system microprocessor restarts the monitor routine as if it had just been initialized.

There are two system microprocessor interrupts which may occur during normal execution of a system command. First is the completion interrupt from the HSS which indicates that an HSS sequence is about to be completed. This interrupt causes an exit to the system microprocessor completion routine. The other interrupt is the data error interrupt from the EDAC which indicates an error has been detected in the data being decoded to make error status information available. This interrupt causes an exit to the system microprocessor data error routine.

5.5 SYSTEM MICROPROCESSOR: COMPLETION ROUTINE

When the HSS is about to finish an instruction sequence, it notifies the system microprocessor by issuing a completion interrupt. Upon receiving the interrupt, the system microprocessor enters the completion routine, as flow charted in figure 5-6. The first function of the routine is to decide if the system command has been completed or if the HSS must repeat the instruction sequence at least once more. To repeat the HSS sequence, the system microprocessor re-enables the HSS and returns to the monitor routine.

Should execution of the system command be completed, the system microprocessor updates the system status to indicate command finished. Then the error counter is checked. If the error counter is not zero, the system microprocessor updates the system status word to reconfiguration. The external system then allows reconfiguration of the memory system, if feasible. If reconfiguration is allowed, the system microprocessor then enters the error processing routine. After completing error processing, or if no reconfiguration was done, the system microprocessor interrupts the power kernel, to indicate that there is time for a system microprocessor CPU test. The power kernel controls the executing of the CPU test. Should the CPU fail, the power kernel maintains control of the system microprocessor and performs a complete system microprocessor test, as done during system initialization.

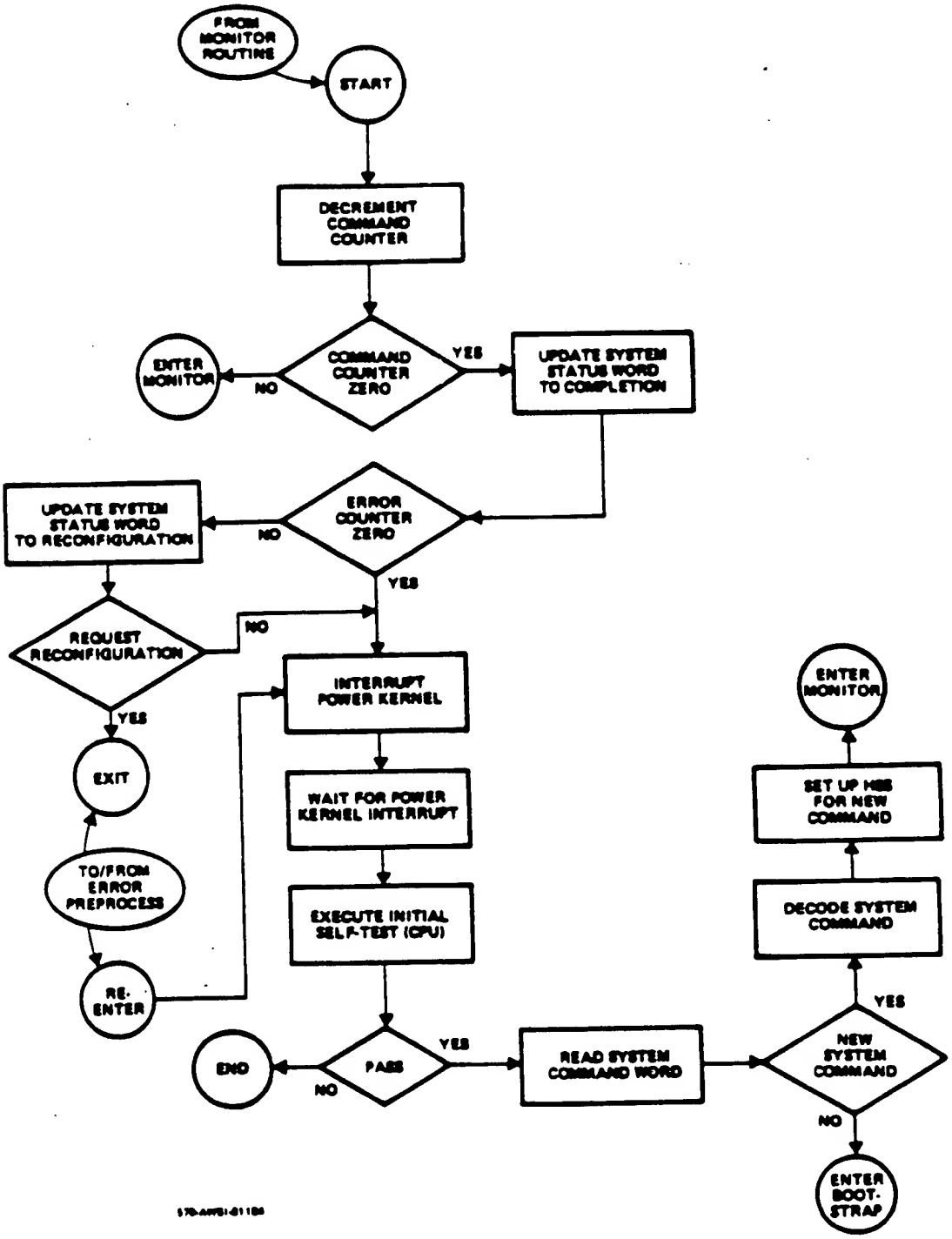


Figure 5-6. System Microprocessor Completion Routine

When the system microprocessor passes the CPU test, it first checks for a new system command. If a new system command has been issued, the command is decoded and the HSS set up to execute the command. The system microprocessor then enters the monitor routine. The system microprocessor performs a bootstrap to test the entire system controller, as in the system initialization, if no new system command has been issued.

5.6 SYSTEM MICROPROCESSOR: DATA ERROR ROUTINE

When a data error occurs during a read operation, the error detecting/correcting circuitry notifies the system microprocessor that an error has occurred. Thus, the error status information can be retrieved by the system microprocessor before it is lost. The EDAC notifies the system microprocesor by issuing a data error interrupt. This interrupt causes the system microprocessor to enter the data error routine which retrieves and stores the error information. Some simple processing is done on the error data to save storage space. The routine is flow charted in figure 5-7.

Upon receiving the interrupt, the system microprocessor disables the interrupts, retrieves the error status information, and then re-enables the interrupts. By not enabling the interrupt logic until after the error status has been retrieved, the system microprocessor guarantees that the error status is not lost due to delays caused by other interrupts. After the error information is retrieved, the error counter is incremented and then the error status is stored away.

When the first error occurs, an error data file is set up to store the error data in a more useful simplified form. If the error is a multiple-bit error, the error status is stored in the multiple bit error file section and the multiple error counter is incremented. Since multiple bit errors are not correctable by the EDAC, the system status word is updated to indicate that bad data exists in the file.

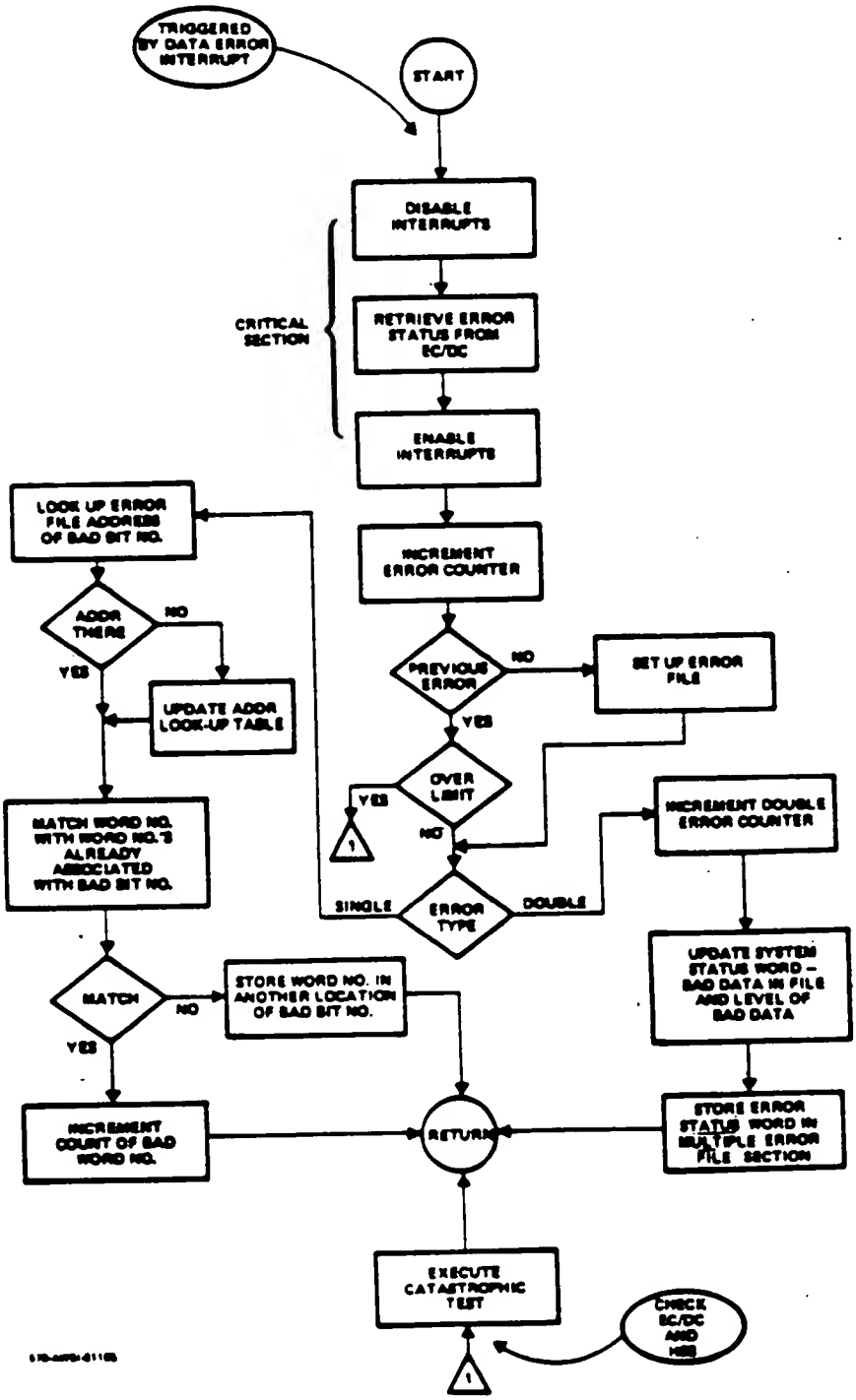


Figure 5-7. System Microprocessor Data Error Routine

Since single-bit errors are corrected by the EDAC, the system status need not be changed when they occur. The location of the faulty bit, bit number, is located and its file area found. In the file area of the bad bit number, the system buffer address word number is also stored. If the word number has been stored in this file area before, the associated counter is incremented to show the number of repeats of this bad bit-word number combination. If the word number has not been stored before, it is loaded with a zero count to show a new bad bit-word number combination. This bit-word number combination is used to decide whether a permanent or a transient failure caused the errors.

Should the error counter indicate more than the allowable number of bits in error, a failure in either the EDAC or the HSS is assumed and a severe test is performed. This severe test halts the system and indicates a halt due to failure in the system status. A test of the EDAC is then initiated. Any faults located by this test are rectified by replacing the bad EDAC modules with standby spares, and the test is restarted. Should this EDAC module replacement not rectify the situation, the system microprocessor assumes the HSS is at fault and conducts a test of it. After the faults have been removed, the file operation is restarted.

5.7 SYSTEM MICROPROCESSOR: ERROR PREPROCESSING ROUTINE

Whenever errors occur during a read operation, the error preprocessing routine, shown in figure 5-8, is entered by the system microprocessor at the conclusion of the read operation. The error preprocessing routine decides the best built-in test for the type of errors recorded.

The first step toward executing a built-in test (BIT) is to reread all the data of the file to remove all transient errors. Any error not exactly duplicated during the reread is considered transient. Should the reread show that no static errors have occurred, a BIT is no longer necessary and the error preprocessing routine is exited. If only multiple bit errors appear permanent, then the file level BIT is executed. This level of BIT tests the entire data file to locate the faults, since multiple bit errors cannot be isolated by the EDAC.

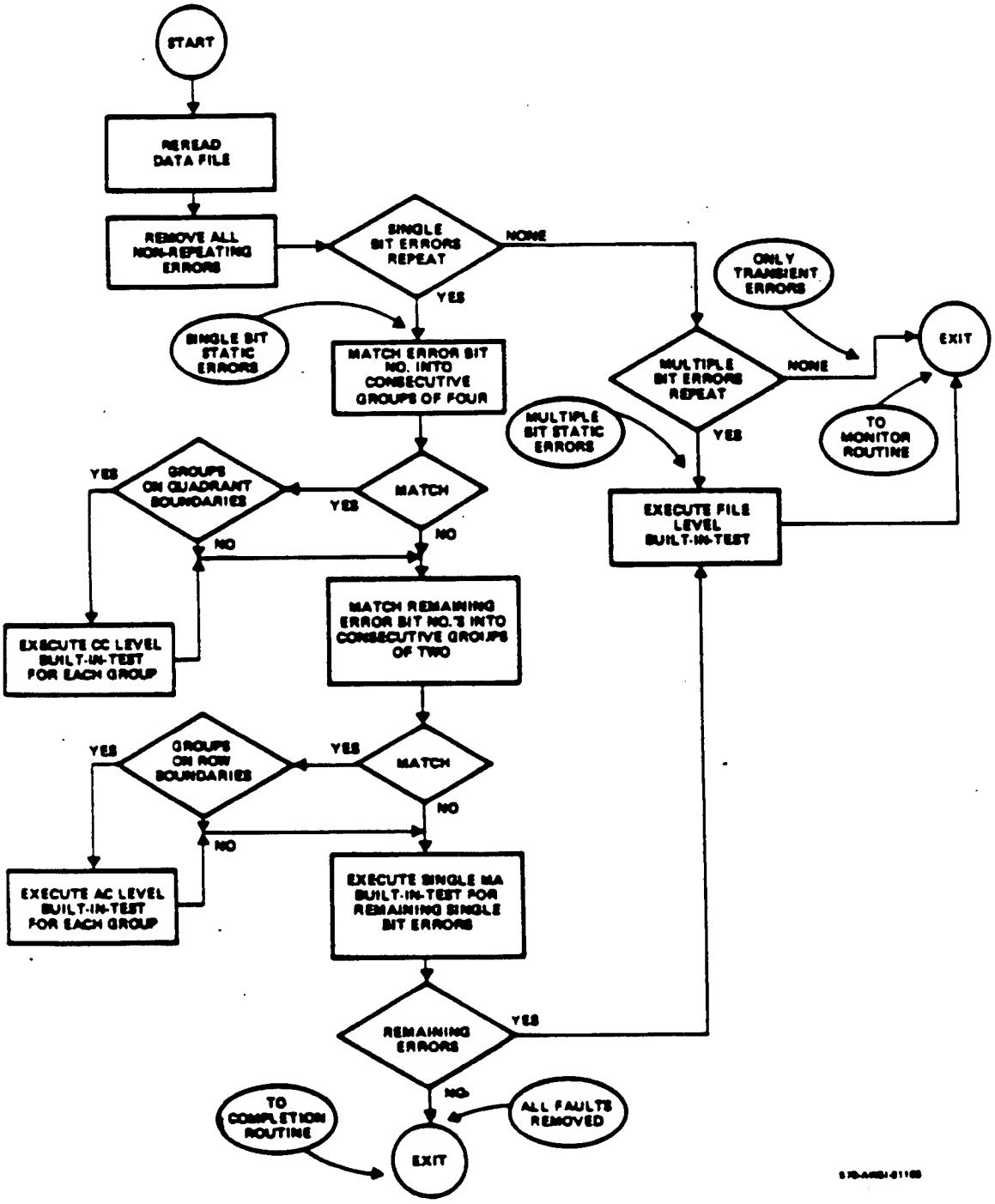


Figure 5-8. System Microprocessor Error Preprocessing Routine

LEX15545

Once the fault areas are located, the system microprocessor reconfigures the affected areas and retests to see if the faults have been corrected. Once the faults are removed, the routine is exited. Should BIT not remove the faults, testing continues until error-free operations are possible.

When single-bit errors are detected, more information is available on the location and nature of the fault, so less drastic BIT's are then possible. The first level of testing is to check for groups of 4 bits which have been in error. These bits, if properly aligned, can signify a channel controller failure, and so the CC-level BIT is executed. Upon completion of the CC-level BIT, the data is reread to discover which faults have been resolved by the CC test. Next, the remaining error bit locations are screened for pairs which are properly aligned to indicate an AC failure. When such bit pairs are found, an AC-level BIT is executed. The faults removed by the AC test are then found and eliminated.

The remaining errors should only be random single-bit errors and their associated coincident multiple-bit errors. For each single-bit error remaining, a memory array-level BIT is executed on the specific memory arrays. Should any faults appear to remain after this final sub-level of testing, a file-level BIT must be executed to resolve the problem.

5.7.1 CC-Level BIT

When the system microprocessor error preprocessing routine determines the type of error detected resembles a channel controller failure, it calls the channel controller (CC) level BIT. The channel controller level BIT is composed of three test sequences: channel controller component test, channel controller comparison test, and replacement test. Together these three test sequences are used to verify that a suspect CC is really faulty, and that replacing it with a spare actually removes the fault. The relationship between the three test sequences is shown in the flow chart of figure 5-9.

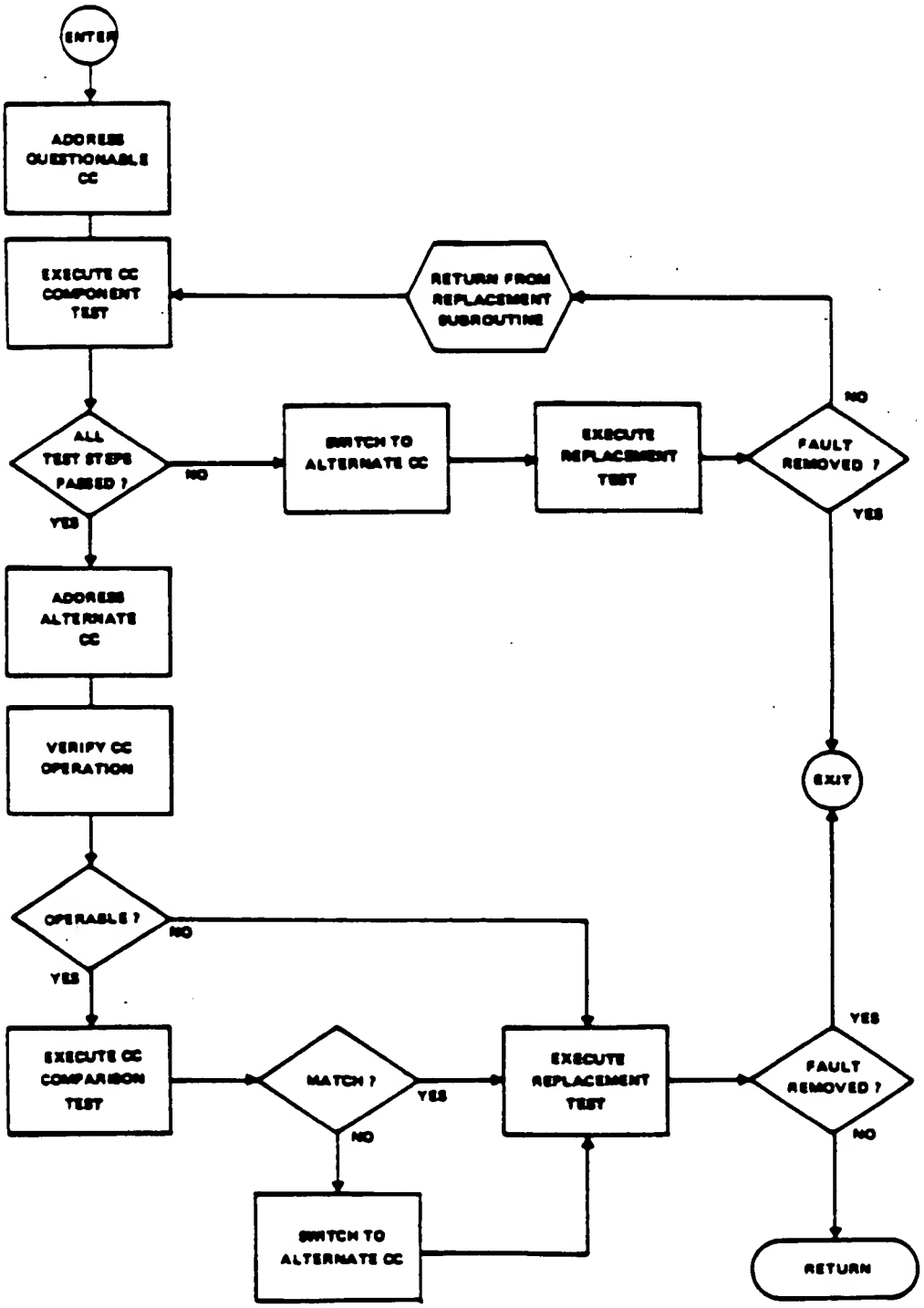


Figure 5-9. CC-Level BIT

The first test sequence of the CC level BIT is the CC component test. This test sequence attempts to verify a CC failure by testing individual components inside the CC. A set of test steps is executed by the CC; each test step verifies one or more functions of a key component. As the CC passes each test step, the function tested is assumed to be operable and can then be used to test another part of the CC, e.g., after the spare row counter (SRC) passes its test steps, it can be used to read the test data from the array controller map (ACM). The individual test steps of the CC component test are listed in table 5-1.

Should the CC fail any of the test steps twice in a row, the CC is assumed to be faulty, indicating that the problem has been found. To verify that the CC alone has been causing the detected data errors, the alternate CC is switched into the quadrant's configuration and its operation is verified by executing the test steps of Table 5-3. The replacement test is then executed. The replacement test is the final check. Should the quadrant pass the replacement test, the quadrant is assumed to be fault free. The test steps of the CC replacement test are detailed in table 5-2.

After the questionable CC passes each test step of the component test sequence, the alternate CC is addressed and the channel controller comparison test sequence is executed. However, before the channel controller comparison test can be executed, the alternate CC must be checked to be sure it is functioning properly. This is done by executing the CC verification test. The CC verification test is a subsequence of the CC component test sequence which performs every test step of the component test except any step which alters nonvolatile memory. Each test step of the verification test is given in table 5-3.

Once the alternate CC has had its operability verified, the CC comparison test is performed. The comparison test compares the contents of each of the NVM address maps. By performing a location-by-location comparison of the maps, and then analyzing every discrepancy, the differences can usually be attributed to a fault in one of the CC's. In this case, the CC whose map is believed to be

TABLE 5-1. CC COMPONENT TEST

1. Address questionable CC and wait for CC acknowledge.
2. Test SRC by incrementing it from one zero test to another, checking the number of increments needed.
3. Test SRC, Y-bus, and M/L muxes by loading known bits from Z-bus into SRC and incrementing SRC to zero as a check on the load.
4. Read the known code words from the special AOM locations into SRC and increment SRC to zero to check the AOM read, and the AOM data retention.
5. Read each of the addresses of the faulty data block, both the row and array pair addresses, from the AOM to the SRC. Increment the SRC to zero for each address, and store values in system microprocessor, checking each for self-consistency.
6. Erase these two words of the AOM and then reread these words into the SRC, incrementing the SRC to zero to verify the erase.
7. Rewrite the addresses back into the AOM by incrementing the SRC to the values stored in the system microprocessor and loading them into the memory latches. Then reread each address back into the SRC, incrementing the SRC to zero to verify the write.
8. Read the value of each AOM location into the SRC, incrementing the SRC to zero for each, and loading the value into the system microprocessor. Then check the self-consistency of each value--duplicate addresses, out-of-bounds addresses, rows listed both as spare and as allocated, etc.
9. Test the temporary address register by incrementing it from one zero test to another, checking the number of increments used.
10. Read the known code words from the active AC address map (AAA) to the temporary address register (TAN) and increment the TAN to zero to check the TAN load, AAA read, and AAA data retention.
11. Using the addresses supplied by the AOM, read the two locations of the AAA used by the bad data blocks into the TAN, incrementing the TAN to zero and storing both values in the system microprocessor. Then erase these two locations of the AAA. After both erasures, read the value of each location into the TAN, incrementing the TAN to zero to verify each erasure.
12. Rewrite these values back into the same locations of the AAA by incrementing the TAN to the values stored in the system microprocessor and then using the addresses supplied by the AOM, perform each write. After rewriting both values, reread the two locations back into the TAN, incrementing the TAN to zero to verify each write.
13. Read the contents of each AAA location into the TAN, incrementing the TAN to zero after each read, storing the values in the system microprocessor. These values are then checked if in bounds.
14. Set and then clear the power bit, latching the zero test to verify the power bit operation.

TABLE 5-2. CC REPLACEMENT TEST

1. Address alternate CC and wait for CC acknowledge.

2 Verify operation of alternate CC.

3 Read the value of each AOM location into the SRC, incrementing the SRC to zero for each, and storing each value in the system microprocessor. Then check the self-consistency of these values.

4. Compare each location of the two AOM's stored in the system microprocessor, noting and storing the differences between the two compared values. Analyze the complete set of differences to determine the type of fault causing the difference and which CC is at fault.

5. Read the value of each active AC address location into the TAN, using the addresses supplied by the AOM. Increment the TAN to zero for each value, storing each value in the system microprocessor. Check each value for validity.

6. Compare each value of the two active AC addresses stored in the system microprocessor, noting and storing any differences. Then analyze the set of differences to determine both the type of fault and the CC causing the difference.

TABLE 5-3. CC VERIFICATION TEST

1. Address CC and wait for CC acknowledge.

2. Test SRC by incrementing it from one zero test to another, checking the number of increments used.

3. Test SRC and Y-bus and M/L muxes by loading known bits from X-bus into SRC, both MSH and then LSH, incrementing SRC to zero as a check on the load.

4 Read the known code words from their special AOM locations into SRC, and increment the SRC to zero to check the AOM read and the AOM data retention.

5. Test the TAN by incrementing it from one zero test to another, checking the number of increments used.

6. Read the known code words from the active AC address to the TAN and increment the TAN to zero to check the TAN load, active AC address read, and active AC address data retention.

7. Set and clear the power bit, watching the zero test to verify the power bit operation.

called. The array controller level BIT is as similar to the CC-level BIT as the AC hardware is to the CC hardware. The AC level BIT consists of three test sequences, as did the CC level BIT: AC component test, AC comparison test, and replacement test. Together these three tests verify that a questionable AC is faulty, and that replacing it actually removes the fault. The way the three test sequences cooperate is flow charted in figure 5-10.

The first difference between the AC level BIT and the CC level BIT is since a CC must be addressed before an AC can, the CC of the quadrant is addressed and verified. Once the operability of the CC has been verified, the AC under question executes the AC component test steps. The AC component test sequence is basically the same as the CC component test; the individual test steps are given in table 5-4. Should the AC fail to pass any of the test steps, an alternate AC is switched into the quadrant configuration and the replacement test is executed to verify that replacing the AC removes the fault.

Once the AC completes the AC component test without failing any test step, an alternate AC is addresed and the comparison test (table 5-5) is executed. Should the analysis of the AC comparison test fail to indicate clearly an AC failure, the third AC is addressed and the comparison test is repeated. With three complete sets of values, the AC comparison test can then indicate which AC is faulty. Then the replacement test can be executed with a good AC and removal of the fault can be verified (table 5-6).

Should the replacement test indicate that replacing the questionable AC does not remove the fault, the error preprocessing routing will then determine whether the CC level BIT or the memory array level BIT is executed next. When less than three AC's are available for the comparison test, the comparison test using only two AC's is used. Should the comparison test not be possible because there are no spares available, the burden of removing the fault is placed upon the replacement test under control of the system microprocessor error preprocessing routine.

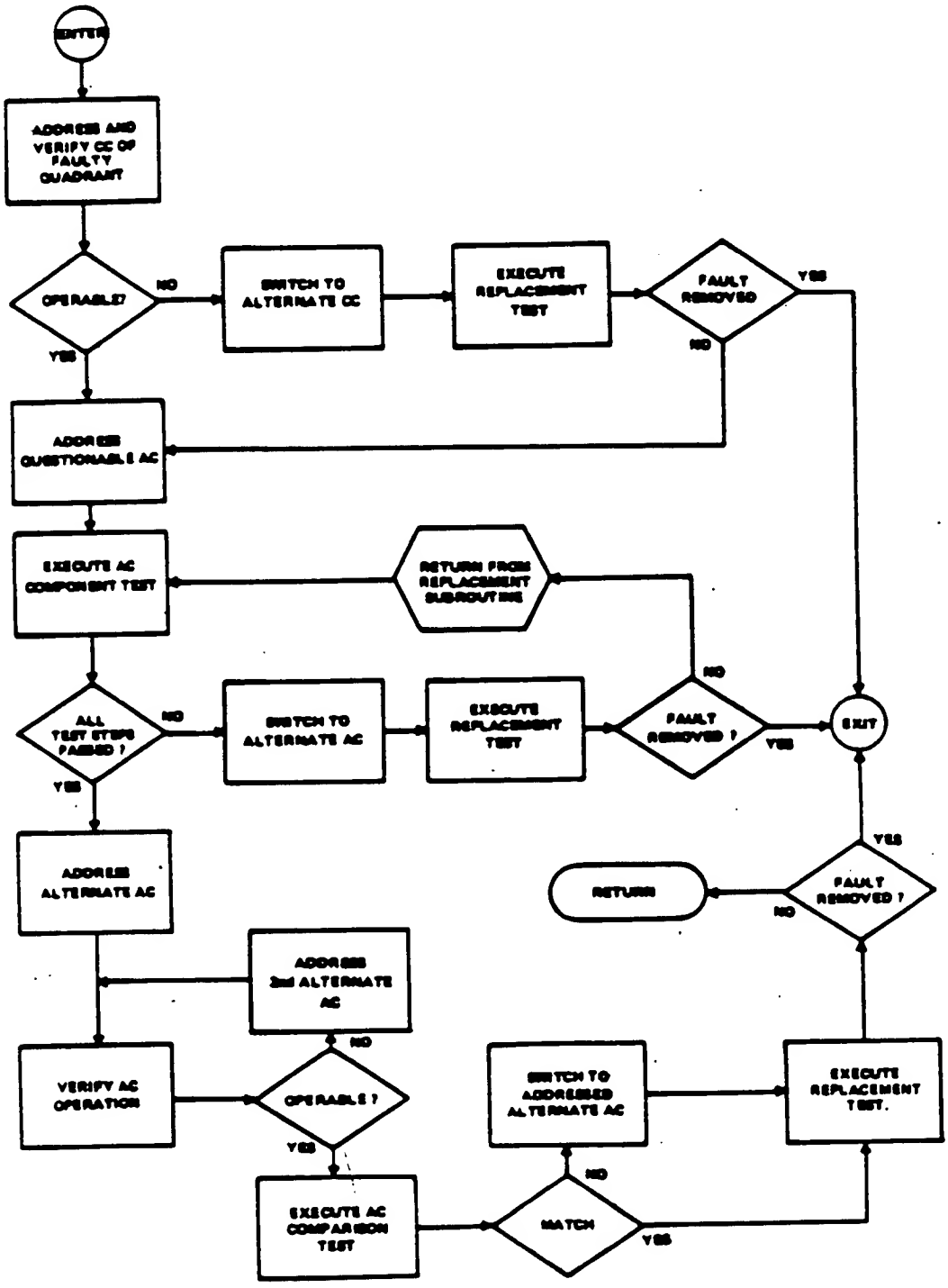


Figure 5-10. AC-Level BIT

TABLE 5-4. AC COMPONENT TEST

1. Address questionable AC and wait for AC acknowledge.

2 Test spare array counter by incrementing it from one zero test to another, checking the number of increments used.

3. Read the known code words from their special locations in the array map into the spare array counter, and increment the spare array counter to zero for each to check the spare array counter load, the array map read, and the array map data retention.

4. Read each of the addressed used by the faulty data block from the array map to the spare array counter. Increment the spare array counter to zero for each address, storing the values in the system microprocessor, and checking each value for selfconsistency.

5 Erase these two words of the array map, then read each into the spare array counter, incrementing the spare array counter to zero to verify the erase.

6 Rewrite the values back into the array map by incrementing the spare array counter to the values stored in the system microprocessor and perform each write. Then reread each value back into the spare array counter, incrementing each to zero to verify the write.

7 Read the value of each array map location into the spare array counter, incrementing the spare array counter to zero for each and storing each value in the system microprocessor. Then check the self-consistency of each value--duplicate addresses, addresses out-of-bounds, addresses listed as both allocated and spare, etc.

TABLE 5-5. AC COMPARISON TEST

1. Address next alternate AC and wait for AC acknowledge.

2. Verify operation of alternate AC.

3. Read the value of each array map location into the spare AC counter, incrementing the spare AC counter to zero for each and storing each value in the system microprocessor. Then check the self-consistency of these values.

4. Compare each value of the two array maps stored in the system microprocessor, noting and storing the differences between each pair of compared values. Analyze the complete set of differences, both the type of fault and which is at fault.

5. Should analysis of the array map discrepancies not yield a clear failure mode, the third AC is addressed and the comparison of array map values is repeated. This third set of values is analyzed to identify which AC is faulty.

TABLE 5-6. AC VERIFICATION TEST

1. Address AC and wait for AC acknowledge.
2. Test spare AC counter by incrementing it from one zero test to another, checking the number of increments used.
3. Read the known code words from their special locations in the array map into the spare AC counter, and increment the spare AC counter to zero for each to check the spare AC counter load, the array map read, and the array map data retention.

5.7.3 MA-Level BIT

Should the system microprocessor error preprocessing routine determine the type of error detected is most likely caused by a memory array failure, the memory array level BIT is executed. The memory array level BIT is composed of the memory array component test sequence and the replacement test sequence, as shown in figure 5-11.

Before an MA can be accessed, the CC and AC of its quadrant must be accessed, so both a CC and an AC are addressed and operationally verified before the MA component test sequence is executed. The MA component test sequence not only checks the operation of the MA itself, but also checks the operation of each of the eight data lines. The check of the data lines is accomplished by shifting data in and out of the MA shift registers eight times, once over each data line. Should the data returning over one of the data lines differ from the data of the others, that data line is assumed to be bad. This data line failure is then verified by switching to another row and repeating the test. From this repeated test, the location of the data line fault can be determined and the action to be taken (abandonment of a row or quadrant) confirmed. The individual test steps of this sequence are listed in tables 5-7 and 5-8.

Whenever an MA which contains data is addressed as the result of a test, every attempt is made to save the data stored there. This begins with the original reread of the faulty data performed by the error preprocessing routine. Each

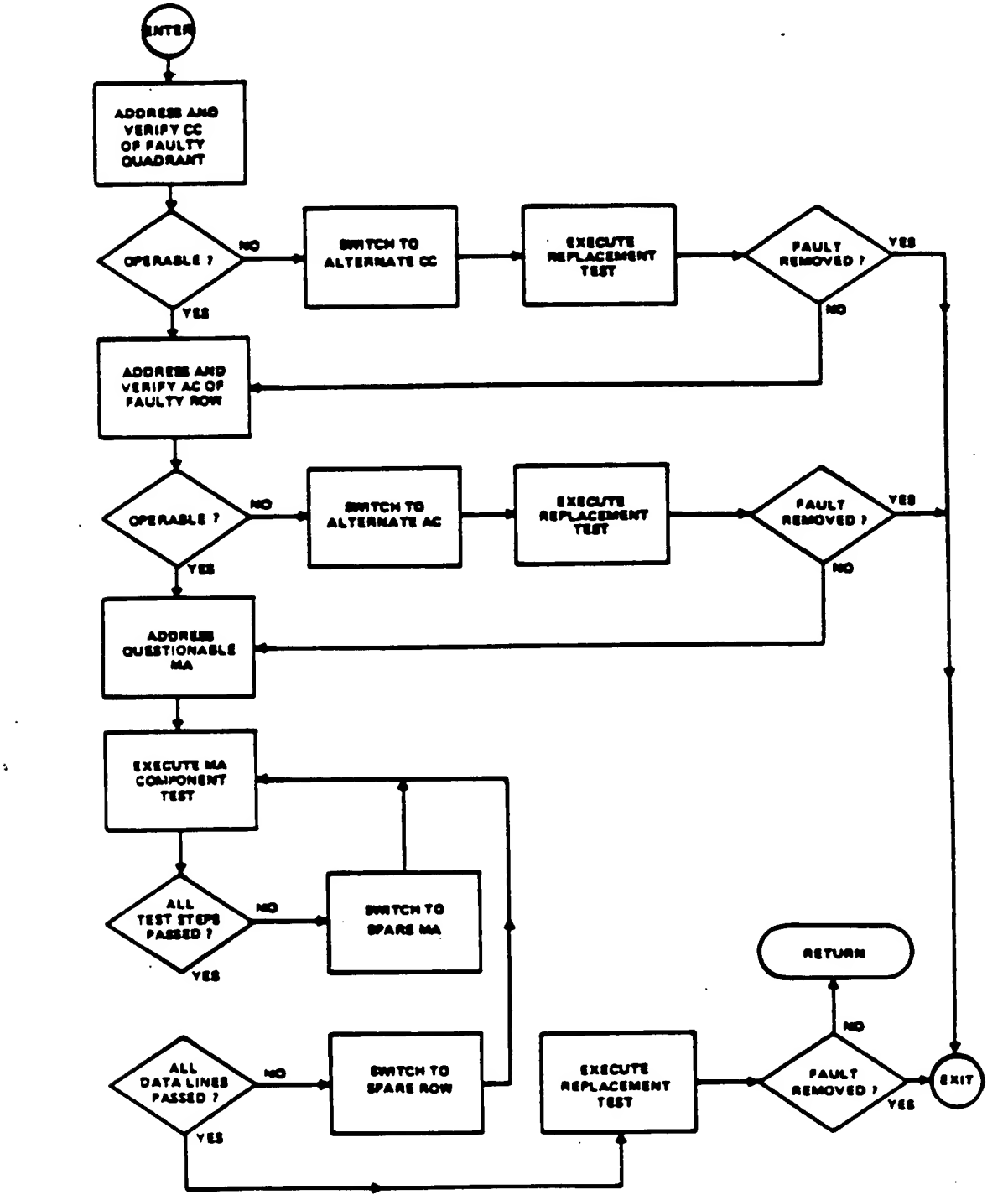


Figure 5-11. Memory Array Level BIT

TABLE 5-7. MA COMPONENT TEST

1. Address questionable MA and wait for MA acknowledge.
2. Load word address register with all ones and then all zeros and then all ones again, watching the zero test signal to verify zero loading.
3. Repeatedly load the word address register with only one bit set to one, shifting the location of the set bit between every load and watching the zero test line to verify each bit is set.
4. Shift sixty-four known data bits into the MA shift registers, and then shift the data back out of the shift registers, checking to make sure the same data returns.
5. Readdress the CC, AC, and MA of the quadrant, changing the data lines used each time. After each readdressing, shift the same known data in and out of the MA shift registers watching to be sure the same results are received after each shift test step.
6. Should any data stored in the MA not be stored in the system microprocessor, attempt to read the data out of the MA and store it in the system microprocessor RAM, checking the data to be sure that it is reasonable.
7. Erase the MA and then read the contents of the MA to verify the erasure.
8. Write a known test pattern into the MA and then read the test data out of the MA to verify the write.

TABLE 5-8. VERIFICATION TEST

1. Address MA and wait for MA acknowledge.
2. Load the word address register with all ones and then all zeros and then all ones again, watching the zero test signal to verify zero loading.
3. Repeatedly load the word address register with only one bit set to one, shifting the location of the set bit between every load and watching the zero test line to verify that every bit gets set.
4. Shift sixty-four known data bits into the MA shift registers and then shift the data back out of the shift registers, checking to make sure the same data returns.
5. Readdress the CC, AC, and MA of the quadrant, changing the data lines used each time. After each readdressing, shift the same known data in and out of the MA shift registers watching to be sure the same results are received after each shift test step.

set of bits which are detected to be erroneous are stored by the system microprocessor as they are corrected. During each subsequent reread of the file, these bits are stored again should the removal of a fault provide additional correctability. Once a faulty MA is replaced by a good spare, the data is stored into this MA and the file data is once again error-free. The only way that data is lost is if the errors in a file are too numerous or the number of iterations through the test sequence is large enough to exceed the amount of system microprocessor RAM available.

Should the MA component test sequence and the replacement test sequence be unable to succesfully verify and remove the faulty MA, the error preprocessing routine then determines whether the CC level BIT or the AC level BIT is executed next.

5.7.4 File-Level BIT

Should a full iteration of the CC level, AC level, and MA level BIT cycle not succeed in removing fault from a data block, or should there not be enough information to isolate a single data block, the file level BIT is executed. The file level BIT is the ultimate effort in fault removal, and as such is not executed except when absolutely necessary. The file level BIT is the only test which generally destroys data stored in MA's; this is because it tests so many MA's that the data cannot be stored in the RAM available. An alternate technique of storing data in spare MA's may be used to save this data.

The file level built-in-test uses all the test sequences of the other built-in-tests, but generally relies on the replacement test sequence flow charted in figure 5-12. The replacement test sequence accesses each component, CC, AC's, and MA's, of a data block and verifies each one after addressing it. The verification test sequences are detailed in tables 5-3, 5-6, and 5-8. Should a component of the data block not pass the verification test, the replacement test switches to the nearest spare and repeats the test sequence. If there is no spare available or the available spare cannot be verified, the appropriate component test is executed to determine whether a row or quadrant must be

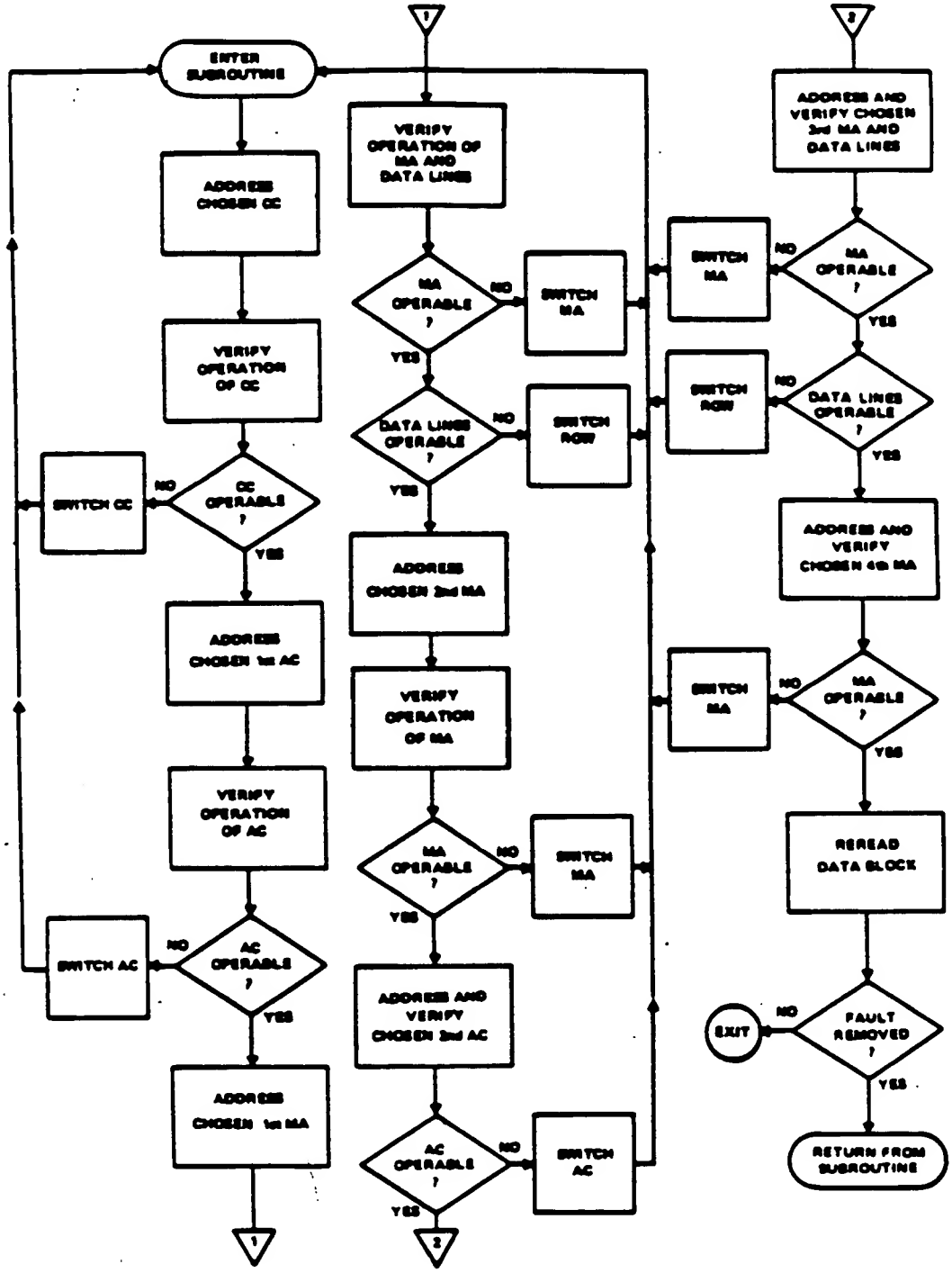


Figure 5-12. Replacement Test Subroutine

switched. After all the components of the data block have been addressed and verified, the data is reread from the data block, whether it is actual data or test data, and the data is checked for faults. If no data errors are detected, the BIT of that data block is complete. Should errors be detected, the built-in testing continues with another level of BIT.

When the replacement test is called by any BIT except the file level BIT, some of the components of the data block have already been operationally verified. These are not verified again by the replacement test. The file level BIT, however, always verifies every component as it is addressed. Should a component fail the verification tests during a file level BIT, the appropriate component test is immediately called and that component thoroughly checked out. Proceeding in this manner, the file level BIT checks the components of an entire file, data block by data block until all the faults are removed.

5.8 HIGH-SPEED SEQUENCER: BASIC SEQUENCE

The high-speed sequencer supplies the detailed commands for the memory wafers to execute each system command. Each sequence is basically a variation of the main sequence, which is flow charted in figure 5-13. The variations of the sequence are selected by the starting address of the sequence and the HSS system microprocessor status, which are supplied by the system microprocessor. Each branch decision in the flow chart represents a test of the corresponding status bits.

First the HSS retrieves the addressing information necessary to operate on a particular data file. The address file, corresponding to the data file, is retrieved from the wafers and stored in the address buffer. The HSS-system microprocessor status is then checked to see which addressing sequence is required. Depending on the status bits, any from one MA in one quadrant to 8 MA's in two quadrants can be addressed. When less than a full quadrant is to be addressed, the status flags also indicate which specific MA or MA pair is to be addressed. Full dual quadrant addressing was discussed in section 4.3.3.

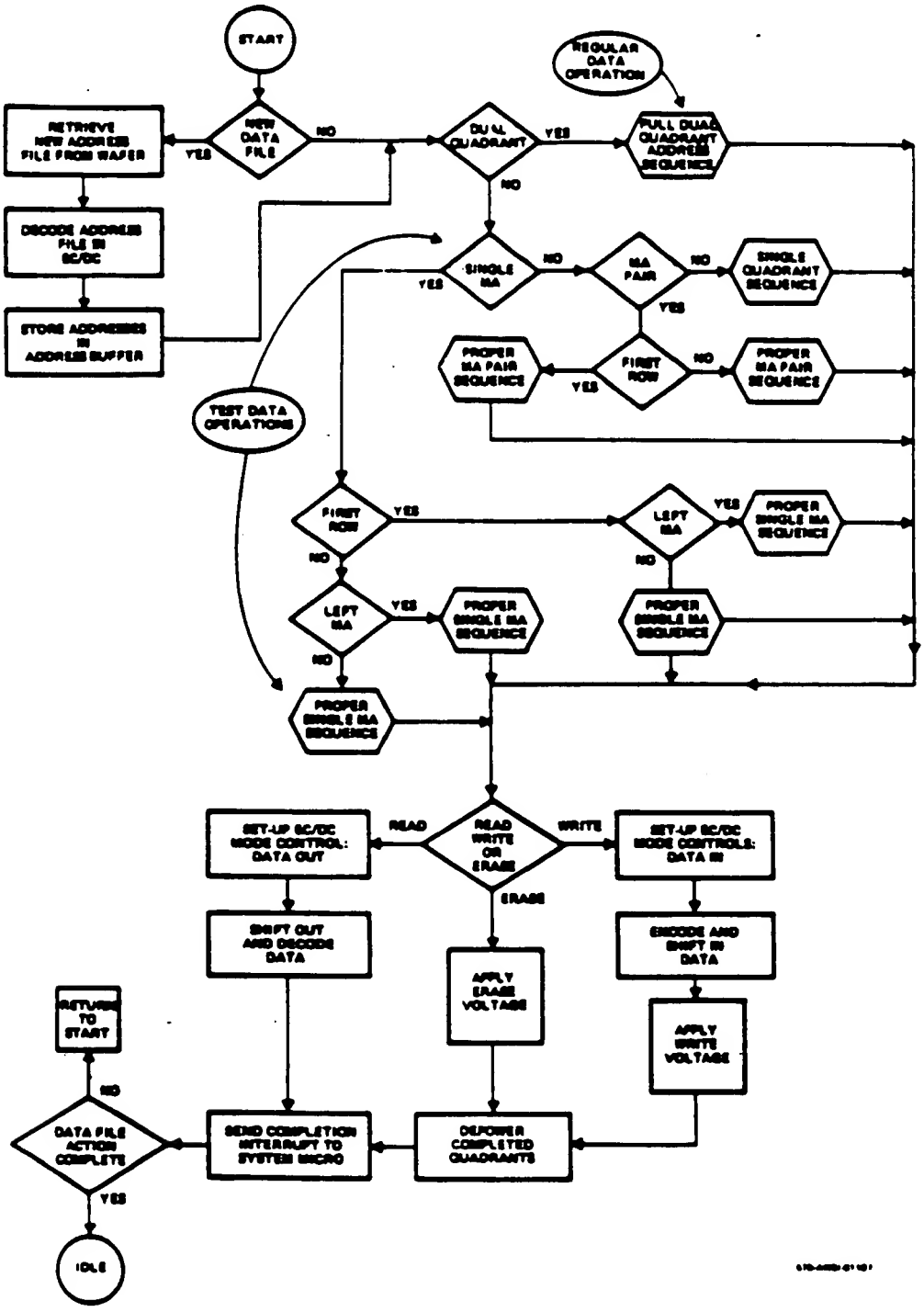


Figure 5-13. High-Speed Sequencer Basic Sequence

After the addressing sequence is chosen and executed, the type of data operation is selected by testing another status flag. For read or write file operations, the EDAC mode is set up, followed by the sequence to shift the data between the addressed MA's and the system interface, via the EDAC. After completing the shifting sequence, the write voltage is applied to the MA's, which assume the low power write state. Since erase operations do not involve data shifting, the HSS powers down the EDAC and then applies the erase voltage to the addressed MA's. After a write or erase voltage has been applied to the addressed MA's, any MA's which have finished their write or erase are addressed and powered down.

When the read, write, or erase sequence has finished, the HSS issues a completion interrupt to the system microprocessor. The system microprocessor then instructs the HSS to either repeat the whole sequence or idle until the next system command is initiated.

Exhibit

A High-Density NAND EEPROM with Block-Page Programming for Microcomputer Applications

YOSHIHISA IWATA, MASAKI MOMODOMI, TOMOHARU TANAKA, HIDEKO OODAIRA, YASUO ITOH, RYOZO NAKAYAMA, RYOUHEI KIRISAWA, SEIICHI ARITOME, TETSUO ENDOH, RIICHIRO SHIROTA, KAZUNORI OHUCHI, AND FUJIO MASUOKA, ASSOCIATE MEMBER, IEEE

Abstract—A 5-V-only CMOS 4-Mb NAND EEPROM with high-speed block-page programming circuits and on-chip test circuits for evaluating the NAND-structured cell is described. This high-density EEPROM has successfully demonstrated the applicability of these techniques for microcomputer applications, which require a large nonvolatile memory system with low power consumption.

I. Introduction

MOST modern computer systems have CPU registers, a random access main memory, and a sequential access secondary memory to achieve high performance and low cost per bit. The necessary features of the secondary memory are large capacity, nonvolatility, and low cost per bit. The rotating magnetic disk, in its variations, is the most popular secondary memory device and has been in existence for some time. Because the magnetic disk needs a rotating mechanism (e.g., electronic motor) and a position sensing mechanism, the computer system with the magnetic disk systems is large and heavy.

High-density EEPROM's are attracting much interest in systems designers who wish to eliminate the rotating magnetic disks in portable computers and battery-powered laptops. Since the conventional EEPROM cell occupies a large area, the EEPROM's cost per bit is high and its capacity is not enough to be used as secondary memory.

Because the NAND EEPROM has a very small cell area, it can realize high density and low cost. The NAND EEPROM is the most promising candidate to replace the magnetic disk system [1]–[3].

At first, this paper introduces the NAND-structured cell and 4-Mb NAND EEPROM and describes the novel circuit techniques used in this EEPROM, which are a block-page mode for high-speed programming and several test modes to realize a highly reliable NAND EEPROM [4]–[7].

Manuscript received August 22, 1989; revised December 5, 1989.
The authors are with the ULSI Research Center, Toshiba Corporation, 1, Komukai, Toshiba-cho, Saiwai-ku, Kawasaki 210, Japan.
IEEE Log Number 8934071.

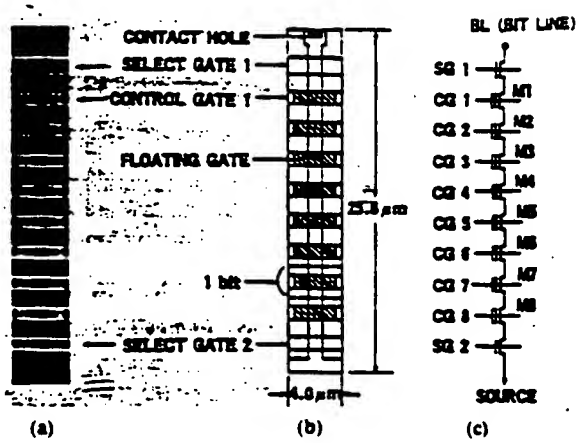


Fig. 1. Top view of the NAND-structured cell. (a) SEM micrograph. (b) Layout. (c) Equivalent circuit.

II. 4-Mb NAND EEPROM

A. NAND Structured Cell

Fig. 1(a)–(c) shows an SEM micrograph, the layout, and the equivalent circuit of the NAND-structured cell for the 4-Mb NAND EEPROM, respectively. As shown in this figure, this NAND-structured cell arranges eight elemental memory cells in series sandwiched between two select gate transistors. Select gate 1 (*SG*1) ensures selectivity and select gate 2 (*SG*2) prevents the cell current from passing during a programming operation. The memory cell transistors are made in a self-aligned double-polysilicon technology. The floating gates are made of first- (lower) layer polysilicon. The control gates are made of second- (upper) layer polysilicon. Both the channel length and width of the memory cell transistors are 1.0 μm.

Fig. 2 shows how the cell size can be reduced by using a NAND-structured cell as compared to the conventional cell (NOR-structured cell). By using 1.0-μm design rules, the memory transistors of both NOR- and NAND-structured cells are illustrated. The NAND-structured cell can realize a

0018-9200/90/0400-0417$01.00 ©1990 IEEE

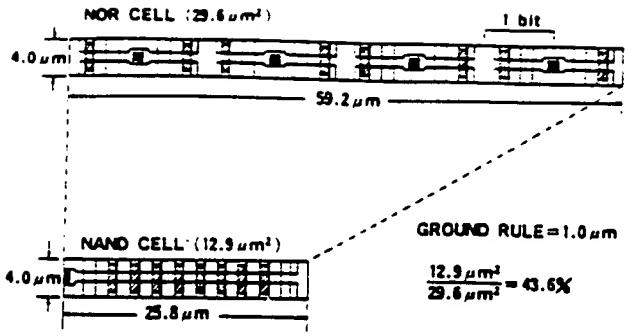


Fig. 2. Comparison of conventional NOR-structured cell with the NAND-structured cell.

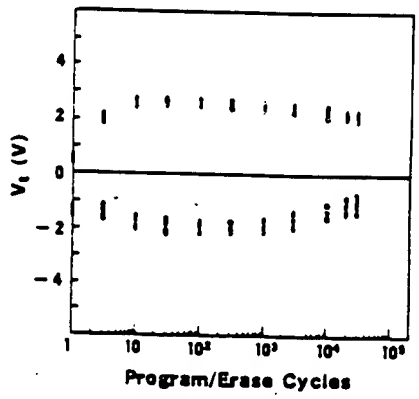


Fig. 3. Endurance characteristic of elemental memory cells of one NAND-structured cell.

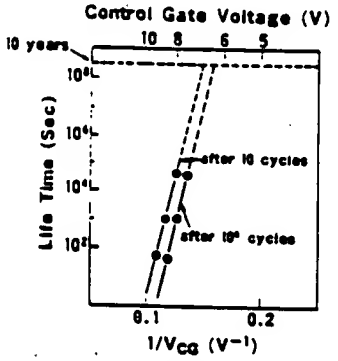


Fig. 4. READ retention characteristic of a elemental memory cell of a NAND-structured cell.

smaller cell area than that of the conventional cell. The whole area of eight NAND-structured cells is 4.0 μm × 25.8 μm and the cell area per bit is 12.9 μm^2. This is only 44% of the area required by a NOR-structured cell.

The endurance of the NAND-structured cell is shown in Fig. 3. Eight elemental memory cells of one NAND-structured cell are measured. The threshold voltage of each elemental memory cell after erasing/programming is independent of the location in the NAND structure. The difference between the threshold voltage of the cell after the tenth erase/program cycle and that of the cell after the 10000th erase/program cycle is very small. But the difference between the threshold voltage of the cell after the first

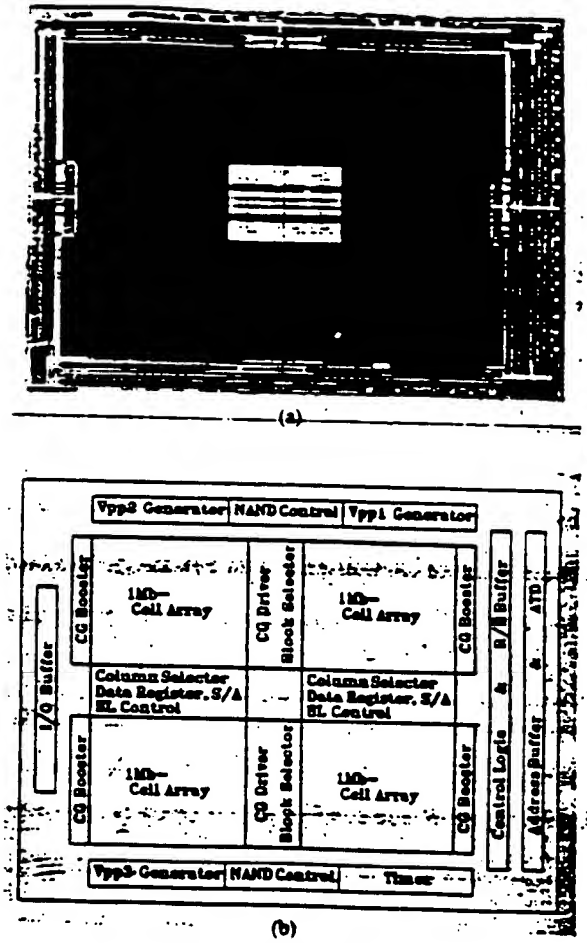


Fig. 5. 4-Mb NAND EEPROM. (a) Photograph. (b) Layout.

cycle and that after the tenth cycle is not small. The warm-up of ten erase/program cycles is needed. The read retention after the 10000th erase/program cycle can be guaranteed for more than ten years in cases where the control gate voltage is below 6.5 V, as shown in Fig. 4.

B. 4-Mb NAND EEPROM

LEX02851

Fig. 5 shows a photograph and the schematic layout of the 4-Mb NAND EEPROM chip. The chip size achieved is 10.7 mm × 15.3 mm. The organization is 512K × 8 b.

The memory cells are divided into four planes. Each plane has 1-Mb (1024 × 1024) cells, in other words 128K NAND-structured cells. The memory cells are divided into 512 blocks for the block-page mode.

The circuit techniques of the 4-Mb NAND EEPROM are described briefly. On-chip high-voltage generators can realize 5-V-only erase/program operations, because erase and program are performed by Fowler–Nordheim tunneling. The block selector drives one pair of 512 pairs of $SG1$ and $SG2$. The CG driver applies 5 V/0 V and V_{GH}/V_{GL} to the control gates in the normal and test modes, respectively. CG boosters raise the control gates of the selected NAND unit to erase/program voltages, which are generated by an on-chip high-voltage generator. The sense amplifier of the

TABLE I
SUMMARY OF KEY ORGANIZATION, TECHNOLOGY, PHYSICAL, AND PERFORMANCE PARAMETERS

ORGANIZATION	512K×8b	
	1Kbyte BLOCK-PAGE MODE	
TECHNOLOGY	N-WELL CMOS	
	TRIPLE-LEVEL POLYSILICON	
	SINGLE ALUMINAM LAYER	
PROCESS PARAMETERS		
CELL	GATE LENGTH	1.0μm
	TUNNEL OXIDE THICKNESS	100A
	INTER-POLY THICKNESS	250A
	(capacitive equivalent oxide thickness)	
PERIPHERAL	GATE LENGTH(NMOS)	2.0μm
	GATE LENGTH(PMOS)	2.5μm
	GATE OXIDE THICKNESS	400A
CELL SIZE	12.9μm^2	
NAND CELL SIZE(8bit)	4 × 25.8μm^2	
CHIP SIZE	10.7 × 15.3mm^2	
ACCESS TIME	1.6μsec	
POWER CONSUMPTION	100mW	
STANDBY POWER	50μW	

dynamic sensing scheme ensures sensing against the small worst-case READ current. The address transient detector (ATD) generates the trigger signal of the dynamic sensing operation. Each bit line has a data register and a bit-line control circuit for block-page programming. Timers and an address buffer, which includes an address latch, can realize a self-timed erase/program cycle which frees the CPU to perform other tasks during erasing/programming. The $R/\bar{B}$ buffer is the output circuit of the $R/\bar{B}$ page signal for indicating one erase/program period and of the $R/\bar{B}$ block signal for indicating the whole block-page mode period.

C. Process and Device Technologies

The memory cells are fabricated by a 1-μm self-aligned double-polysilicon technology. The tunnel-oxide thickness is around 100 Å. The inter-polysilicon dielectric is an oxide–nitride–oxide (ONO) stack with a capacitive equivalent oxide thickness of 250 Å.

The peripheral circuits are fabricated by a 2-μm triple-polysilicon n-well CMOS process. The gate oxide thickness is 400 Å. The gate lengths of NMOS and PMOS are 2.0 and 2.5 μm, respectively.

The main device parameters are shown in Table I.

III. BLOCK-PAGE MODE

Fig. 6 shows a block diagram of the 4-Mb NAND EEPROM. The 4-Mb cells are split into 512 blocks. One block consists of 1024 NAND cell units. One block is the least unit for erasing/programming.

One block-page cycle consists of one block simultaneous erase cycle and eight successive page program cycles. Fig. 7 shows one block cell array and its control circuits. At first,

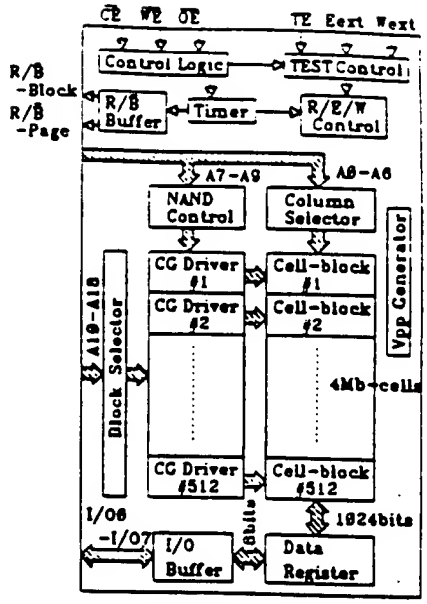


Fig. 6. Block diagram of the 4-Mb NAND EEPROM.

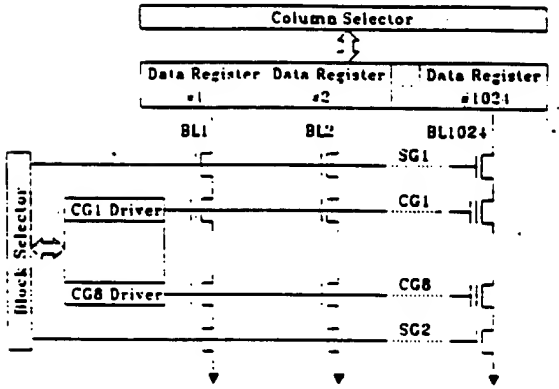


Fig. 7. One block cell array and its control circuits.

all cells in the selected block are erased at once by applying 17 V to all control gates of the selected block and 0 V to the 1024 bit lines. Electrons are injected into all the floating gates of the cells. The threshold voltage of the erased cell becomes the enhancement mode of approximately 2 V. All cells in the block are erased simultaneously.

After the block erase operation is accomplished, the page programming starts from the source side cells, which are connected with the *CG*8 line, to the bit-line side cells, which are connected with the *CG*1 line, successively, in order to prevent interference between programmed cells and cells being programmed. One page-program cycle consists of a 1024-b data load sequence followed by a page program sequence. After the data load sequence, each of the bit lines is raised to programming voltages in accordance with the logic state of its data register.

At first, the programming starts at the source side cells which are connected with *CG*8. Zero volts are applied to the control gate of the selected cells (*CG*8) while 22 V are applied to the control gates of the unselected cells

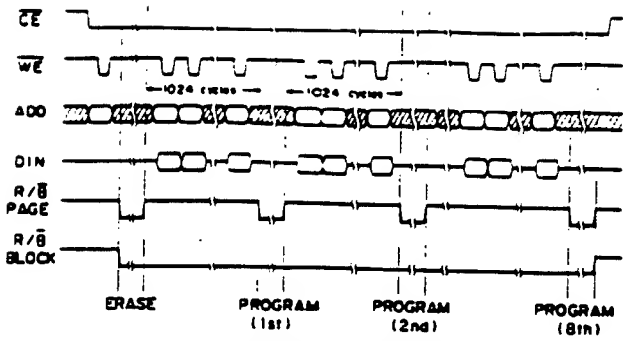


Fig. 8. Switching waveforms of block-page mode.

($CG1$–$CG7$), which are closer to the bit line than the selected cells. These unselected cells act as pass transistors.

In the case of ZERO programming, the bit line is raised to 22 V and electrons are emitted from the floating gate to the bit line by the electric field between the bit line and the floating gate of the selected cell. The threshold voltage of the selected cell is pushed into the depletion mode of approximately −3 V.

In the case of ONE programming, the bit line is raised to 11 V. But no electrons are emitted from the floating gate to the bit line, because the electric field between the bit line and the floating gate of the selected cell is insufficient to initiate tunneling. The threshold voltage of the selected cell remains at 2 V.

After programming the cells connected with $CG8$, the programming of the cells connected with $CG7$ starts. Zero volts are applied to the control gate of the selected cell ($CG7$) and the control gate of the unselected cell ($CG8$), which is closer to the source line than the selected cell ($CG7$). Twenty two volts are applied to the control gates of the unselected cells ($CG1$–$CG6$), which are closer to the bit line than the selected cell ($CG7$). The bit line is raised to 22 V in the case of ZERO programming, and 11 V in the case of ONE programming.

Subsequently, the programming continues from the source side cell to the bit-line side cell successively. Typical erasing time per block is 1 ms. Typical programming time per page, including the data load sequence, is 1 ms. As a result, the total programming time of one block page is 9 ms.

Fig. 8 shows the switching waveforms of the block-page mode.

After $\overline{CE}$ goes down to "low," first $\overline{WE}$ toggle triggers a block erase operation. Following the erase operation, eight page program operations are performed. Input data are stored in data registers corresponding to $\overline{WE}$ toggles. After $\overline{WE}$ stays at "high" level for around 1 ms, this EEPROM interprets that the data load sequence has finished. Each of the stored data is transferred to each of the bit lines; 1024 bits of data are programmed to the selected 1024 cells.

This EEPROM has the data registers, address input latch circuits, and timer circuit for self-timed erase/program. The EEPROM also has two extra output signals: the $R/\overline{B}$ page output signal indicates one erase/program period and the $R/\overline{B}$ block output signal indicates the whole block-page period.

If the CPU host monitors these signals, the CPU is free of controlling this EEPROM except for the data load sequence. The data registers, address input latch circuits, timer, and $R/\overline{B}$ output signals make interface circuits simple and raise CPU efficiency.

IV. Test Mode

Tests for a EEPROM cell, for example, the endurance test and the data retention test, are important for realizing a highly reliable EEPROM. But these tests are time-consuming.

This 4-Mb NAND EEPROM has several test modes for shortening test time and raising test efficiency. When the TE of the extra control input is "high," this EEPROM is in a test mode.

A. Chip Erase and all Block-Page Program

When address input $A0$ is raised to V_{HH} for a positive voltage between 9 and 15 V while TE is "high," the chip erase mode is initiated and 4-Mb cells are all erased at once.

The all block-page mode is identical to the one block-page mode except that address input $A13$ is raised to V_{HH} while TE is "high." Then, 512 blocks are block-page programmed at the same time.

The chip erase mode and the all block-page mode are used to shorten the test time when carrying out tests that require a large number of erase/program cycles, such as the device endurance test.

B. Cell Threshold Voltage Measure Mode

In the NAND EEPROM, the unselected cell acts as a pass transistor in READ/program operation. So, the threshold voltage of the ONE programmed cell must be controlled in the range between 1 and 3 V. In this section, the technique of measuring the threshold voltage of a ONE programmed cell is described.

Fig. 9 shows the typical READ current as a function of the control gate voltage of the selected elemental memory cell of the NAND-structured cell. In this case, the control gate voltage of the unselected elemental memory cells of the NAND-structured cell are 5 V. The threshold voltage of the ONE programmed cell is around 2 V. The threshold voltage of the ZERO programmed cell is around −3 V.

In the normal READ operation, the control gate voltage of the selected cell is 0 V, and that of the unselected cell is 5 V. So, the unselected cell can act as a pass transistor even if this cell is ONE programmed. If a ZERO programmed cell is selected, the selected NAND-structured cell generates READ current. If a ONE programmed cell is selected, the NAND-structured cell does not generate READ current.

Now, in the test READ mode, a control gate voltage can be externally applied. V_{GL} is applied to the control gate of

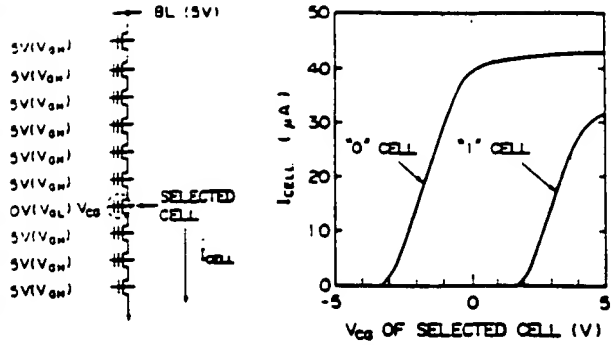


Fig. 9. Cell current as a function of the control gate voltage on the selected cell of the NAND cell.

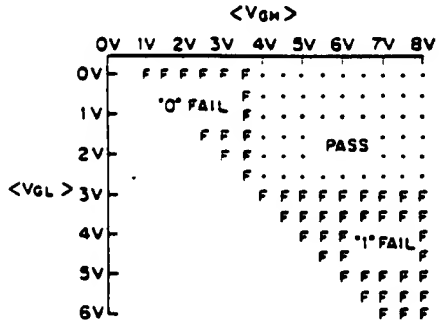


Fig. 10. Schmoo plot of test READ mode.

the selected cell. V_{CH} is applied to the control gates of the unselected cells.

V_{GL} is applied over 2 V. So, the NAND-structured cell generates READ current even if the ONE programmed cell is selected. If V_{GH} is below 2 V, the unselected ONE programmed cell does not act as a pass transistor. So the NAND-structured cell does not generate READ current even if the ZERO programmed cell is selected.

Fig. 10 shows a schmoo plot in the test READ mode. The ZERO fail area shows that V_{GH} is smaller than the threshold voltage of the unselected ONE programmed cell. So cell current does not flow in spite of the ZERO programmed cell selected. The ONE fail area shows that V_{GL} is larger than the threshold voltage of the selected ONE programmed cell. So cell current flows in spite of the ONE programmed cell selected.

Thus, the threshold voltage of a ONE programmed cell can be easily evaluated from this schmoo plot. In this case, the threshold voltage of the ONE programmed cell is about 3 V.

By using this mode, the deviation in threshold voltage of a ONE programmed cell, between the initial state and the post-stressed state, can be measured. Fig. 11 shows the distribution of the threshold voltage of the ONE programmed cell in the 4-Mb NAND EEPROM. The distribution of the threshold voltage after ten erase/program cycles is between 1.6 and 3.2 V. The distribution after 10000 erase/program cycles is between 1.2 and 2.8 V. The deviation between the 10th and 10000th cycles is so small

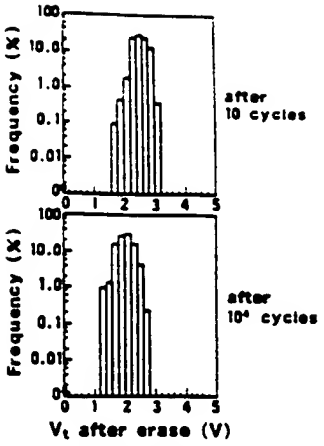


Fig. 11. Distribution of ONE programmed cell of the 4-Mb NAND EEPROM.

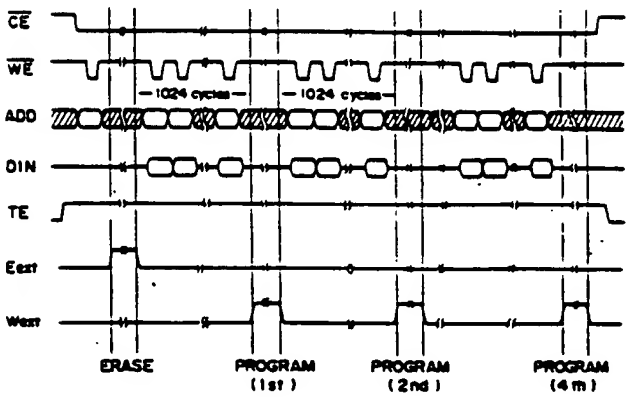


Fig. 12. Switching waveforms of test erase/program mode.

that this 4-Mb EEPROM reliably performs after 10000 erase/program cycles.

C. External Timer and External Erase/Program Voltage

This NAND EEPROM has two extra control inputs for evaluating the NAND-structured cell. While *TE* is "high," erase time and program time are controlled by the E_{ext} and W_{ext} inputs, respectively. If $\overline{CE}$ transits from "low" to "high" during an erase/program cycle in a block-page mode, the next program cycle cannot be initiated in the same block-page mode cycle.

Switching waveforms of the test erase/program mode are shown in Fig. 12.

On-chip high-voltage generators have output pads. From these output pads, erase/program voltages can be externally applied variably.

The features of the NAND-structured cell about erase/program can be evaluated by using these modes.

V. Application

LEX02854

Fig. 13 shows floppy disks attached to a typical computer system and Fig. 14 illustrates NAND EEPROM's attached to a similar system. The system with floppy disks needs a floppy disk drive (FDD) that has a rotating

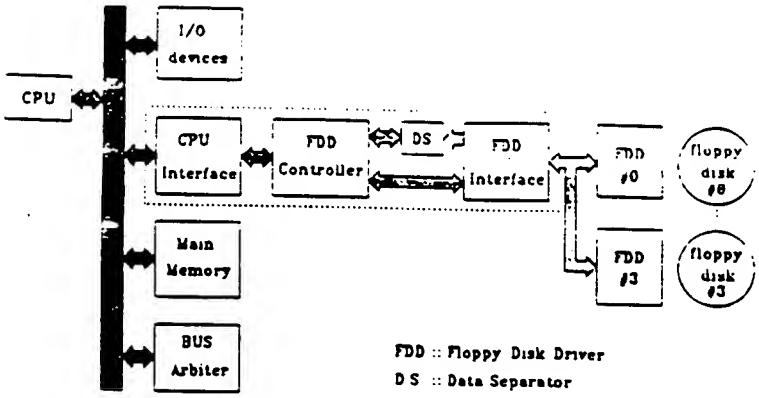


Fig. 13. Block diagram of computer system with floppy disks.

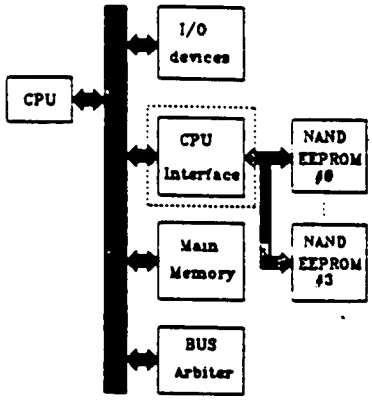


Fig. 14. Block diagram of computer system with NAND EEPROM's.

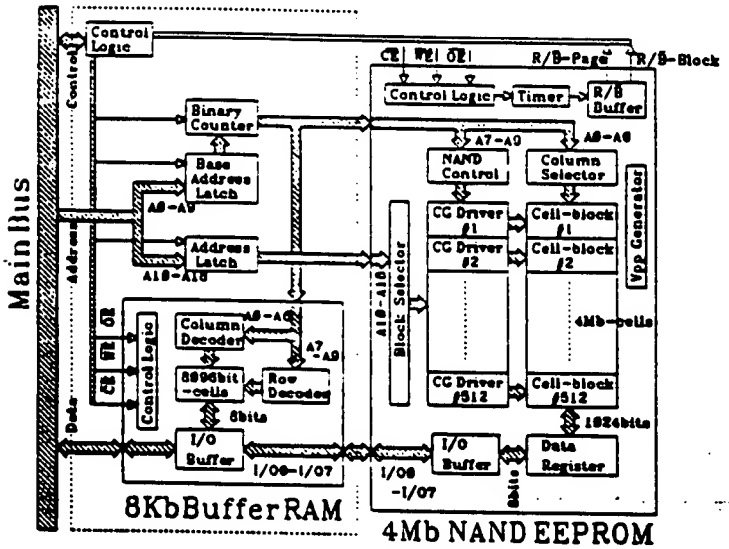


Fig. 15. Block diagram of the NAND EEPROM's and interface.

LEX02855

mechanism and a position sensing mechanism. An interface unit consists of FDD interface, a data separator (DS), an FDD controller, and CPU interface. DS extracts valid data from output signals of FDD. CPU interface contains a parallel-serial converter. Thus, this interface unit is complex.

On the other hand, the NAND EEPROM is a tiny semiconductor chip. The NAND EEPROM requires no extra control unit except for the CPU. The system with the NAND EEPROM is simpler, lighter, and less power-consuming than the system with magnetic media for a secondary memory device. Also, the access time per bit or

program time per bit is 1/1000th shorter than that of the magnetic media including the seek time.

To further raise the CPU utilization on this system, a 1-kilobyte buffer RAM and a controller unit are needed for interface to the NAND EEPROM's. The buffer RAM is used for temporary storage of one block data. The controller has an address register for temporary storage of a selected block address and a binary counter for generating page and NAND-control address. By supporting these circuits, the CPU only carries out 1-kilobyte data loading to the buffer RAM during one block-page mode (as shown in Fig. 15).

The NAND EEPROM needs no power to retain its storage data. The EEPROM needs only 5-V power supply. Therefore, the NAND EEPROM is suitable for compact microcomputer applications, which are off-line systems with large storage. The memory card is an especially good application for the NAND EEPROM, whose characteristics are large capacity and nonvolatility with no power supply. This card can be used for personal cards of hospitals and banks, for an electronic camera, for the data cartridge of an electronic musical instrument, and for navigation systems.

VI. Summary

A high-density, 5-V-only CMOS EEPROM with a NAND-structured cell using Fowler–Nordheim tunneling for programming has been realized. The block-page mode is adopted for high-speed programming and easy microprocessor interface. On-chip test circuits for test-time shortening and for cell characteristic evaluation realize highly reliable EEPROM's. The NAND EEPROM has many applications for microcomputer systems that require small size and large nonvolatile storage systems with low power consumption.

References

[1] R. Stewart *et al.*, "A high density EPROM cell and array," in *VLSI Technology Dig. Tech. Papers*, May 1986, pp. 89–90.

[2] F. Masuoka, M. Momodomi, Y. Iwata, and R. Shirota, "New ultra high density EPROM and flash EEPROM with NAND structure cell," in *IEDM Tech. Dig.*, Dec. 1987, pp. 552–555.

[3] R. Shirota *et al.*, "A new cell for ultra high density 5V-only EEPROMS," in *VLSI Technology Dig. Tech. Papers*, May 1988, pp. 33–34.

[4] M. Momodomi *et al.*, "New device technologies for 5V-only 4Mb EEPROM with NAND structure cell," in *IEDM Tech. Dig.*, Dec. 1988, pp. 412–415.

[5] Y. Itoh *et al.*, "An experimental 4Mb CMOS EEPROM with a NAND structured cell," in *ISSCC Dig. Tech. Papers*, Feb. 1989, pp. 134–135.

[6] M. Momodomi *et al.*, "A high density NAND EEPROM with block-page programming for microcomputer applications," in *Proc. CICC*, May 1989, Paper 10.1.

[7] M. Momodomi *et al.*, "An experimental 4-Mbit CMOS EEPROM with a NAND-structured cell," *IEEE J. Solid-State Circuits*, vol. 24, no. 5, pp. 1238–1243, Oct. 1989.

Yoshihisa Iwata was born in Aichi, Japan, on May 16, 1960. He received the B.E. degree in physics from Nagoya University, Japan, in 1983.

He joined the Toshiba Research and Development Center, Kawasaki, Japan, in 1983. Since then he has been working on the circuit design of high-density DRAM's and EEPROM's at the Toshiba ULSI Research Center.

Mr. Iwata is a member of the Institute of Electronics, Information and Communication Engineers of Japan.

Masaki Momodomi was born in Fukuoka, Japan, on March 11, 1958. He received the B.E. degree in electronic engineering from Kyushu University, Fukuoka, Japan, in 1980.

He joined the Toshiba Research and Development Center, Kawasaki, Japan, in 1980. Since then he has been working on the circuit design of high-density DRAM's, EPROM's, and EEPROM's at the Toshiba ULSI Research Center.

Mr. Momodomi is a member of the Institute of Electronics, Information and Communication Engineers of Japan.

Tomoharu Tanaka was born in Yamaguchi, Japan, on July 8, 1962. He received the B.E. and M.E. degrees in applied physics from the University of Tsukuba, Japan, in 1985 and 1987, respectively.

He joined the Toshiba Research and Development Center, Kawasaki, Japan, in 1987. Since then he has been working on the circuit design of high-density EEPROM at the Toshiba ULSI Research Center.

Hideko Oodaira was born in Aomori, Japan, on December 31, 1967. She graduated from Hirosaki Technical High School, Japan, in 1986.

She joined the Toshiba Research and Development Center, Kawasaki, Japan in 1986. Since then she has been working on the circuit design of high-density EEPROM at the Toshiba ULSI Research Center.

Yasuo Itoh was born in Osaka, Japan, on August 1, 1953. He received the B.E. and M.E. degrees in electronic engineering in 1977 and 1979, respectively, and the Ph.D. degree in electrical engineering in 1982, all from Osaka University.

In 1982 he joined the Toshiba Research and Development Center, Kawasaki, Japan. Since then he has been engaged in the research and development of high-density DRAM's and EEPROM's at the Toshiba ULSI Research Center.

Dr. Itoh is a member of the Japan Society of Applied Physics, the Physical Society of Japan, and the Institute of Electronics, Information and Communication Engineers of Japan.